(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) Übersetzung der europäischen Patentschrift

(87) EP 0 257 581 B1

(10) DE 37 88 758 T 2

(51) Int. Cl.[5]: G 06 F 15/80

DE 37 88 758 T 2

(21) Deutsches Aktenzeichen: 37 88 758.0
(86) Europäisches Aktenzeichen: 87 112 152.1
(86) Europäischer Anmeldetag: 21. 8. 87
(87) Erstveröffentlichung durch das EPA: 2. 3. 88
(87) Veröffentlichungstag der Patenterteilung beim EPA: 12. 1. 94
(47) Veröffentlichungstag im Patentblatt: 23. 6. 94

(30) Unionspriorität: (32) (33) (31)
29.08.86 US 902343

(73) Patentinhaber:
International Business Machines Corp., Armonk, N.Y., US

(74) Vertreter:
Teufel, F., Dipl.-Phys., Pat.-Ass., 71155 Altdorf

(84) Benannte Vertragstaaten:
BE, CH, DE, ES, FR, GB, IT, LI, NL, SE

(72) Erfinder:
Li, Hungwen, Pleasantville, N.Y. 10570, US

(54) Polymorphes Maschennetzwerk für Bildverarbeitungssystem.

DE 37 88 758 T 2

B E S C H R E I B U N G

Polymorphes Maschennetzwerk für Bildverarbeitungssystem

HINTERGRUND DER ERFINDUNG

1. Anwendungsbereich der Erfindung

Die Erfindung bezieht sich auf einen Feldprozessor, der aus einem Netzwerk von Verarbeitungselementen besteht, und dabei insbesondere auf ein Feldverarbeitungsnetzwerk, bei dem jedes Verarbeitungselement in einem Feld von Verarbeitungselementen mit einem Verbindungssteuermechanismus, auf den das Programm zugreift, zusammen mit steuerbaren begrenzten Maschennetzzwischenschaltungen zu entsprechenden Verbindungselementen ausgestattet ist, um eine programmierbare Auswahl an Netzwerkkonfigurationen zu bieten.

2. Beschreibung der Technik

Die folgenden Veröffentlichungen spiegeln den Stand der Technik wider:

Veröffentlichte Artikel

1. Sternberg, "Biomedical image processing", Computer, Jan. 1983. Dieser Artikel beschreibt ein Feld zellulärer Automaten von identischen Zellen, die mit ihrem nächsten Nachbar für die iterative Nachbarschaftsverarbeitung von digitalen Bildern verbunden sind. Ein Serpentinenschieberegister konfiguriert seriell Nachbarschaftseingänge für die 3 x 3 Nachbarschaft.

2. Turney und andere, "Recognizing Partially Occluded Parts", IEEE Transactions on Pattern Analysis and Machine Intelligence, Juli, 1985, SS. 410-421. In diesem Artikel werden zahlreiche Techniken aufgezeigt, einschließlich Hough-Transformation und gewichtetes Schablonen-Matching.

3. Mudge und andere, "Efficiency of Feature Dependent Algorithms for the Parallel Processing of Images", IEEE 0190-3918/83/0000/0369 1983, 369-373, zeigt, wie die Architektur eines Bildverarbeitungssystems von einer Konfiguration in Form von zahlreichen Unterbildprozessoren profitieren kann, bei denen die Verarbeitungselemente über eine Art Netzwerk miteinander kommunizieren. Die Autoren erklären den Unterschied zwischen funktionsabhängigen Algorithmen und funktionsunabhängigen Algorithmen.

4. Sternberg und andere, "Industrial Morphology". In diesem Artikel wird die Kombination eines Einzelsystems für die Bildverarbeitung und die Strukturerkennung beschrieben.

5. D.E. Shaw, "The NON-VON Supercomputer", internal report, Columbia University, Aug. 1982. In diesem Bericht wird ein paralleles System mit einer I/O-Umschaltung in jedem Verarbeitungselement und Flag-Registern gezeigt, die einzelne PEs aktivieren und deaktivieren.

6. M.J. Kimmel, R.S. Jaffe, J.R. Mandeville und M.A. Lavin, "MITE: Morphic Image Transform Engine, An Architecture for Reconfigurable Pipelines of Neighborhood Processors, IBM RC11438, 10. Okt. 1985. In diesem Artikel wird ein wiederkonfigurierbares Netzwerk von Verarbeitungselementen beschrieben, die zahlreiche Zwischenschaltungen von PE zu PE über Busverbindungen unter Bedienerkontrolle ermöglichen.

7. A.J. Kessler und J.H. Patel, "Reconfigurable Parallel Pipelines for Fault Tolerance", IEEE, CH1813-5/82/0000/0118, 1982. In diesem Artikel wird eine wiederkonfigurierbare Pipeline-Verbindung für Graceful-Degradation beschrieben.

8. S.R. Sternberg, "Parallel Architecture for Image Processing", IEEE, CH1515-6/79/0000-0712, 1979 zeigt ein PE-Netzwerk mit voller Konnektivität.

9. T.N. Mudge, E.J. Delp, L.J. Siegel und H.J. Siegel, "Image Coding Using the Multimicroprocessor System PASM", IEEE, 82CH1761-6/82/0000/0200, 1982.. In diesem Artikel wird eine Verarbeitungselementzwischenschaltung beschrieben, die durch ein Zwischenschaltnetzwerk hergestellt wird.

10. S.R. Sternberg, "Language and Architecture for Parallel Image Processing", Pattern Recognition in Practice, North-Holland Publishing Co., 1980. Hier wird ein komplexes Netzwerk von PEs sowie dessen Betrieb beschrieben.

11. Z. Chen, CC Change und T.L. Chia, 1985, IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Miami Beach, Florida, 18.-20. November 1985, Seiten 80-87. Hier wird ein wiederkonfigurierbarer zellulärer Feldprozessor beschrieben, der über eine Architektur aus speicherverstärkten Prozessoren verfügt, die über Omega-Netzwerke und einen Algorithmus zur erneuten Konfigurierung des Systems verbunden sind.

12. D.G. Wallbrook und D.J. Woollons Electronics Letters Vol. 7, Nr., 15. Juli 1971, Seiten 385-387. In diesem Artikel wird ein paralleler Verarbeitungscomputer zur Strukturerkennung beschrieben, der aus einem rechteckigen Feld von identischen zwischengeschalteten Verarbeitungsmodulen besteht, die über allgemeine Daten- und Befehls-Highways mit einer Befehlseinheit verbunden sind.

* Die US-Patentschrift 4,174,514, 13. November 1979 zeigt einen Feldprozessor mit benachbarten Verarbeitungselementen, die für den gegenseitigen Zugriff auf Bilddaten in überlappenden Feldern zweier benachbarter Bild-Slices zwischengeschaltet sind.

* Die US-Patentschrift 4,215,401, CELLULAR DIGITAL ARRAY PROCESSOR, 29. Juli 1980, beschreibt einen Feldprozessor, bei dem jede "Verarbeitungselementzelle" über zwei Akkumulatoren

verfügt, die die Zelle (mit Ausnahme der Zellen am Feldrand) mit den beiden Nachbarzellen entlang einer Achse sowie mit den beiden Nachbarzellen entlang der orthogonalen Achse verbinden.

* Die US-Patentschrift 4,380,046, Fung, MASSIVELY PARALLEL PROCESSOR COMPUTER, 12. April 1983, zeigt einen Bildprozessor, der eine räumliche Translation durch vertikales oder horizontales Verschieben oder "Gleiten" von Bits zu benachbarten Verarbeitungselemente und P-Registern zu P-Registern durchführt, sofern dies von einem anderen, sogenannten G-Register erlaubt wird. Jedes Verarbeitungselement umfaßt eine Arithmetik-, Logik- und Routing-Einheit (ALRU), eine I/O-Einheit und eine lokale Speichereinheit (LMU). Die ALRU besteht aus drei wichtigen Komponenten: einem Binärzähler/Schieberegisteruntereinheit, einer logischen Slider-Untereinheit (P-Register) und einer Maskenuntereinheit (G-Register).

* Die US-Patentschrift 4,398,176, Dargel und andere, DATA ANALYZER WITH COMMON DATA/INSTRUCTION BUS, 9. August 1983, beschreibt ein Bildverarbeitungsfeld, bei dem jedes Verarbeitungselement über externe Befehlssteuerleitungen verfügt, um zu prüfen, ob Busdaten als Befehl oder als Daten verwendet werden. Jedes Verarbeitungselement umfaßt auch einen Mechanismus, der aus der Bitstruktur eines Befehls festlegt, ob der Befehl lokal oder global ist, und der die Weiterleitung stoppt, wenn es sich um einen lokalen Befehl handelt.

* Die US-Patentschrift 4,601,055, Kent, IMAGE PROCESSOR, 15. Juli 1986, zeigt einen niedrigstufigen Ikone-zu-Ikone-Prozessor mit Pixel-um-Pixel-Vorwärtstransformation und Pixel-um-Pixel-Rückwärtstransformation.

* Die US-Patentschrift Nr. 3,287,702 bezieht sich auf einen parallelen Netzwerkcomputer mit Verarbeitungselementen, die über Schaltungen verfügen, mit denen die Steuersignale, die von einem zentralen Steuermittel als Antwort auf interne

Bedingungen innerhalb der Verarbeitungselemente empfangen werden, geändert werden können.

Die Verfahren auf dem Stand der Technik ermöglichen eine permanente Konfiguration von Verarbeitungselementen in einer Pipeline oder einem anderem Strukturbildverarbeitungssystem. Nach dem Stand der Technik wird ein Bildverarbeitungssystem über Schaltnetzwerke und Busse wiederkonfiguriert. Gemäß dem Stand der Technik ist ein Bittransfer zu benachbarten Verarbeitungselementen möglich. Bei den Verfahren auf dem Stand der Technik ist jedoch von keiner Erfindung die Rede, die sehr schnelle Schaltung innerhalb des Verarbeitungselements ermöglicht, das als polymorphes Maschennetz programmiert werden kann, um die Konfiguration dynamisch zu den Daten zu optimieren, die in folgenden Konfigurationen verarbeitet werden: Zeichenkette, Maschennetz, Baum, Würfel, Pyramide.

Zelluläre Automaten sind für die Bildverarbeitung, Computer Vision und andere Berechnungen in der Physik sehr nützlich. Alle bestehenden Zwischenschaltnetzwerke für zelluläre Automaten gehen von einer festen Struktur wie beispielsweise einer Zeichenkette, einem Maschennetz, einem Baum, einem Würfel oder einer Pyramide, usw. aus. Jede Struktur eignet sich für bestimmte Arten von Berechnungen, ist jedoch für Berechnungen ungeeignet, die nicht der Struktur entsprechen. Da die Netzwerkzwischenschaltstruktur eingebaut, d.h. festinstalliert ist, kann diese selbst bei Erfassung einer Nichtübereinstimmung nicht geändert werden. Bei Nichtübereinstimmung ist eine geringe Leistungsfähigkeit zu verzeichnen.

Ein NxN-Maschennetz beispielsweise ist eine optimale Verbindung für lokale Operationen bei der Bildverarbeitung. Die Leistung dieses Netzwerks ist bei der Berechnung globaler Operationen gering (es werden z.B. N Zyklen zur Berechnung von MINIMUM, einer linearen Komplexität, benötigt). Auf der anderen Seite ist eine Baumverbindung ideal für die Berechnung von MINIMUM (es werden nur log N Zyklen, eine logarithmische

Komplexität, benötigt). Diese Verbindung leistet bei der Berechnung von lokalen Operationen eines Bildes jedoch nicht viel, da die Nachbarschaftsverbindung fehlt.

Neben der allgemein mangelhaften Leistungsfähigkeit bei der Berechnung im Fall einer Verbindung, die nicht mit dem Algorithmus übereinstimmt, ist die Feststruktur auch beim Entwurf eines Algorithmus unflexibel. Dies liegt vor allem bei den Beschränkungen im Datenfluß. Bei der Zeichenkettenverbindung beispielsweise geht der Datenfluß nur in eine Richtung, d.h. von links nach rechts. Eine solche Einschränkung begrenzt auch den Algorithmusbereich: nur Algorithmen mit einem "Zeichenkettendatenfluß" können vom "Zeichenkettennetzwerk" profitieren. In dieser Hinsicht sind die festen Netzwerke nur für bestimmte Zwecke geeignet und haben einen engen Anwendungsrahmen.

Ein weiterer Nachteil der festen Verbindungsstruktur besteht darin, daß ikonische und Zwischenverarbeitung nicht gleichzeitig leistungsstark unterstützt. Dieser Nachteil ist vor allem bei einer wichtigen Anwendung von zellulären Automaten augenfällig, nämlich bei der Computer Vision, bei der sowohl ikonische (oder Bild-) Verarbeitung als auch die Transformation von ikonischen zu symbolischen Daten (genannt Zwischenstufenverarbeitung) zwei integrale Bestandteile sind. Aus diesem Nachteil ergibt sich ein I/O-Problem, da das Bild nach der ikonischen Verarbeitung außerhalb des Netzwerks für die weitere Zwischenverarbeitung gesendet werden muß.

ZUSAMMENFASSUNG DER ERFINDUNG

Gegenstand der Erfindung ist es, eine schnelle Konnektivität mit Programmzugriff auf jedes Verbindungselement in einem Feld zu ermöglichen, so daß die Verarbeitungselemente auf effektive Art und Weise programmierbar gruppiert werden können, ohne den Kostenaufwand für komplexe Verbindungen zu haben, die für universelle Konnektität benötigt werden.

Ein weiterer Gegenstand der Erfindung ist es, eine angemessene programmierbare Steuerung der Konnektivität jedes Verarbeitungselements in einem Feld zu bieten, so daß die Verarbeitungselemente von Zeit zu Zeit programmierbar umgruppiert werden können, und zwar unter der adaptiven Steuerung eines Computers, der den Bedarf nach einer optimierten Umgruppierung feststellt, oder unter Aufsicht eines Bedieners, der den Bedarf einer optimierten Umgruppierung vorsieht.

Ein weiterer Gegenstand der Erfindung ist es, eine externe Speicherdatenverbindung zum Verarbeitungselement zu ermöglichen, bei dem eine bestimmte Hardware umgangen werden kann und zusätzliche Operationsflexibilität erreicht wird.

Eine Funktion der Erfindung gemäß den Ansprüchen besteht in der Verwendung eines Feldes von polymorphen Maschennetzverbindungselementen, die jeweils über Verarbeitungsfähigkeiten verfügen, die in einer Steuer- und Recheneinheit mit einem Speicher integriert sind, und des weiteren über programmierbare Verbindungssteuerfähigkeiten mit geographischen Zielverbindungen und logischen Verbindungen verfügen.

Eine weitere Funktion der Erfindung gemäß den Ansprüchen besteht in der programmierbaren Kurzschlußfähigkeit im polymorphen Maschennetzverbindungselement. Mit der Kurzschlußfähigkeit kann eine Reihe von auftretenden Verarbeitungselementen als Drahtverbindungen bei der Datenübertragung von einem sendenden PE zu einem ferngelegenen PE ohne Zyklusverzögerung verwendet werden.

Eine weitere Funktion der Erfindung gemäß den Ansprüchen bezieht sich auf ein "polymorphes Maschennetzwerk", das ein zusammengesetztes Netzwerk aus einem herkömmlichen Maschennetz extern zu jedem PE und einem internen Netzwerk innerhalb jedes PE ist, um Standardstrukturen und andere neue nützliche Strukturen über Software-Steuerung unterzubringen, so daß die Verbindung der Berechnung angepaßt werden kann. Durch die

"polymorphe" Funktion kann das Netzwerk sich selbst adaptiv "umformen", um ein flexibles Algorithmus-Design zu ermöglichen und größere Anwendungsbereiche abzudecken.

Eine spezielle Funktion im Zusammenhang mit Computer Vision unterstützt die Zwischenstufenverarbeitung (ikonische zu symbolische Transformation) durch das polymorphe Maschennetz, so daß eine effiziente Architektur entsteht, bei der I/O-Probleme vermieden werden.

Eine weitere besondere Funktion der Erfindung gemäß den Ansprüchen bezieht sich auf Flag-Register in den Verarbeitungselementen, die für bedingte Operationen, einschließlich Wiederkonfiguration für adaptive Selbstoptimierung und Fail-Soft-Fähigkeit benutzt werden.

Eine weitere nützliche Funktion der Erfindung gemäß den Ansprüchen bezieht sich auf eine begrenzte Anzahl von Multibit-Strukturregistern, die jeweils auf eine begrenzte Anzahl von Hardware-Strukturen zugreifen, die über die Software ausgewählt werden können und die besonders bei zellulären Automaten nützlich sind, wobei die Strukturen vom polymorphen Maschennetz gebildet werden können. Zu diesen Strukturen gehören Busse, mehrere Bäume, Würfel und Pyramiden, die jeweils optimal für entsprechende Berechnungsarten sind und über einen Crossbar-Schalter als Antwort auf eine Bitstruktur in einem ausgewählten Strukturregister ausgewählt werden können.

Ein Vorteil der Erfindung liegt in der hohen Durchlaßgeschwindigkeit zu relativ niedrigen Kosten, die mit Hilfe programmierbarer begrenzter Konnektivität innerhalb des Verarbeitungselements erreicht wird, um die Optimierung einer Feldverbindung von Verarbeitungselementen physisch und elektronisch zu ermöglichen.

Ein weiterer Vorteil besteht darin, daß die Ergebnisse der

Zwischenstufenverarbeitung im Zusammenhang mit der Programmierung verwendet werden können, um eine adaptive Wiederkonfiguration effizient als Verbindungsfunktion zwischen der Programmierung und den Zwischenergebnissen der Verarbeitung zu ermöglichen. Das bedeutet, daß die Zwischendaten verwendet werden können, um eine adaptive Optimierungsumgruppierungsfunktion zu aktivieren sowie eine adaptive Selbstoptimierung und Fail-Soft-Fähigkeiten.

Ein weiterer Vorteil liegt in der Einfachheit und dem zweidimensionalen Aspekt der Erfindung, bei der zahlreiche Zwischenschaltverarbeitungselemente auf einer kleinen Zahl von Chips in Feldern angeordnet werden können.

Die oben genannten Aspekte und andere Gegenstände, Funktionen und Vorteile der Erfindung gemäß den Ansprüchen gehen aus der nachfolgenden Beschreibung eines bevorzugten Ausführungsbeispiels der Erfindung anhand der Begleitzeichnungen noch näher hervor.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

FIG. 1 ist eine Systemblockdiagramm eines Bildprozessors, der aus einem Netzwerk von polymorphen Maschenverarbeitungselementen besteht, wobei ein Verarbeitungselement als Funktionsblockdiagramm gezeigt wird.

FIG. 2 ist ein Diagramm der Schaltfähigkeit eines Verbindungssteuermechanismus CCM eines polymorphen Maschenverarbeitungselements.

FIG. 3 ist ein Funktionsblockdiagramm des Verbindungssteuermechanismus eines polymorphen Maschenverarbeitungselements.

FIG. 4 ist ein Diagramm, das die Bildung linearer Zeichenkettenfelder von polymorphen Maschenverarbeitungselementen zeigt.

FIG. 5 ist ein Diagramm, das die Bildung der Reihenbäume von polymorphen Maschenverarbeitungselementen zeigt.

FIG. 6 ist ein Diagramm einer anderen Ansicht der Reihenbäume.

FIG. 7-10 sind vereinfachte Diagramme, die einen Chipbereichsvergleich und eine Kommunikationsentfernungsverbesserung zeigen, die sich aufgrund der Verwendung der polymorphen Maschenverbindungselemente ergeben haben.

FIG. 11 ist ein Diagramm mit Umkehrreihenbäumen von polymorphen Maschenverarbeitungselementen.

FIG. 12-20 sind Diagramme mit ausgewählten Feldkonfigurationen, die durch die programmierbare Verbindung von polymorphen Maschenverbindungselemente leicht hergestellt werden können.

FIG. 21 ist ein genaues Blockdiagramm eines einzelnen polymorphen Maschenverarbeitungselements entsprechend dem bevorzugten Ausführungsbeispiel der Erfindung.

BESCHREIBUNG EINES BEVORZUGTEN AUSFÜHRUNGSBEISPIELS DER ERFINDUNG

FIG. 1 zeigt ein polymorphes Maschennetzwerk für ein Bildverarbeitungssystem, das aus einem MxM Fled 1 von Verarbeitungselementen 2 unter Steuerung eines Host-Computers H 3 besteht. Jedes Verarbeitungselement hat eine begrenzte Menge von Verbindungen; im bevorzugten Ausführungsbeispiel sind vier Verbindungen vorhanden, eine Verbindung zu jedem der orthogonalen (nicht-diagonalen) Nachbarn. Diese orthogonalen Nachbarn werden als Kartesische Nachbarn bezeichnet. Die orthogonalen Verbindungen werden für das Verarbeitungselement 4 mit den direktionalen Idenfikationen NESW gekennzeichnet. Diese Verbindungen sorgen dafür, daß der Ausgang eines beliebigen Verarbeitungselements direkt zu einer begrenzten Anzahl von Nachbarn (vier im bevorzugten Ausführungsbeispiel) geführt

wird. Die allgemeine Programmier- und Housekeeping-Steuerung wird über Bus 9 vom Host-Computer 3 durchgeführt.

Ein Verarbeitungselement wird genauer gezeigt. Das Verarbeitungselement 5 wird vergrößert gezeigt, um Platz für die internen Organe ALU 6, MEM 7, CCM 8 sowie die NESW-Verbindungen zu haben.

Jedes Verarbeitungselement (PE) im bevorzugten Ausführungsbeispiel verfügt über vier Kartesische Verbindungen. Die Kartesischen Verbindungen werden aus praktischen Erwägungen als NESW bezeichnet. Sie sind mit den benachbarten PEs in der angegebenen Richtung verbunden. Dieses einfache Maschennetz von Kartesischen Verbindungen ist ohne den CCM 8 in der Lage, Verbindungen zum benachbarten PE herzustellen. Dies ist bei der Bildverarbeitung eine wichtige Funktion. Diese einfache Maschenvorrichtung ist für die Technologie mit sehr hohem Integrationsgrad (VLSI) geeignet.

Nicht-Kartesische PEs sind für die direkte Verbindung nicht angeschlossen. Diagonale Verbindungen und Fernverbindungen sind bei Drähten oder Metallisierung nicht möglich. Diese Verbindungen komplizieren den Herstellungsprozeß; Drahtbündel führen zu Masse und Entfernung mit der inhärenten Lichtgeschwindigkeitsverzögerung.

Im bevorzugten Ausführungsbeispiel wird auf die Nicht-Kartesischen PEs über auftretende Kartesische PEs durch die programmierte Steuerung ihrer entsprechenden CCM 8-Fähigkeiten zugegriffen. Die Struktur der CCM 8-Verbindung sieht so aus, daß der Eingang zum gewünschten Ausgang kurzgeschlossen wird. Die Verbindung kann das Muster eines Turms im Schach (gerade Kartesisch mit optionaler Erweiterung) oder eines Pferdes (Kartesisch X, Kartesisch Y = = remote off-Cartesian) haben, jedoch nicht das Muster des Läufers (diagonal mit optionaler Erweiterung). Komplexe Routing-Strukturen können eingerichtet werden. Ein komplexes Routing zu einem Nicht-Kartesischen Ziel

kann so eingerichtet werden, daß eine große Anzahl von PEs ohne Zyklusverzögerung durchlaufen wird.

Als Alternative zu den einfachen Kartesischen Verbindungen im bevorzugten Ausführungsbeispiel kann die begrenzte Menge der PE-PE-Verbindungen auch Verbindungen zu diagonal benachbarten PEs umfassen. Direkte Verbindungen zu sehr weit entfernten PEs sind jedoch zu komplex, da diese Verbindungen unter der Programmsteuerung entsprechend der Erfindung durch Kombinationen von Kartesischen oder anderen einfachen Verbindungen hergestellt werden können.

FIG. 2 ist ein Diagramm der Schaltfähigkeit des Verbindungssteuermechanismus CCM 8, der in FIG. 1 gezeigt wurde. Die wichtigste Funktion ist der Betrieb als Schaltnetzwerk, um eine der NESW-Verbindungen in der X-Querachse mit einer der NESW-Verbindungen in der Y-Querachse mit Hilfe der Bitwerte in einem Strukturregister zu verbinden. Im bevorzugten Ausführungsbeispiel wird eine Auswahl zwischen den Strukturregistern getroffen, wobei die Auswahl des Strukturregisters von einem Bitwert in einem Strukturauswahlregister gesteuert wird. FIG. 2 liegt in Diagrammform vor und zeigt keine Details der eigentlichen Hardware. Im bevorzugten Ausführungsbeispiel werden nicht alle Verbindungen, die in der 4x4 Matrix zur Verfügung stehen, auch benötigt, obgleich alle verfügbar sind. Die Matrix 10 hat Eingänge 11 in Y-Dimension mit Verbindungen zum Leiter 12 in X-Dimension. Wie gezeigt wurde dies eingerichtet, um den N Leiter 13 über die Verbindungselektronik 14 und 15 sowie den Schnittpunkt 16 mit der E-Verbindung 17 zu verbinden. Andere Verbindungen zu den SWN-Verbindungen 18, 19 und 20 können alternativ oder gleichzeitig aktiviert wird. Die Steuerung liegt bei den Bitwerten des Strukturregisters 21 oder den Bitwerten des Strukturregisters 22. Die Auswahl des Strukturregisters erfolgt durch das Strukturauswahlregister 23. Aus dem Schrägstrich der Verbindungsleitung zu 16 geht hervor, daß die Strukturregister 21 und 22 jeweils 16 Bits haben. Verbindungen zu den Verbindungsleitungen 24, 25 und 26

sind ebenfalls möglich. Der Crossbar-Schalter 10 stellt eine oder zahlreiche der 16 Verbindungen her, die von den Bitwerten in den ausgewählten Strukturregistern 21 oder 22 gesteuert werden.

Jedes Verarbeitungselement ist mit einem oder mehreren seiner Nachbarn verbunden, je nach Steuerung der Bitwerte in einem der beiden Strukturregister. Ein sofortiger Wechsel der Verbindung kann einfach durch Umschalten auf das andere Strukturregister erfolgen. Die Auswahl der Strukturregister wird von den Bitwerten in einem einfachen binären Strukturauswahlregister vorgenommen. Zum besseren Verständnis kann ein Strukturregister als Standardstrukturregister bzw. alternatives Strukturregister angesehen werden, auf das ein 1- oder 0-Wert im binären Strukturauswahlregister zugreift.

FIG. 3 ist ein Funktionsblockdiagramm des CCM 8 von FIG. 1, der die in FIG. 2 beschriebenen Schaltfähigkeiten implementiert und bestimmte andere Richtungs- und Logikfunktionssteuerungen durchführt. Eine vereinfachte Logikverbindungs-Capability 30 wird für AND-, OR-, XOR (Exklusiv OR)- und ANDALLBIT-Capability gezeigt. Es stehen noch andere Capabilities zur Verfügung, mit der Komplexität erhöhen sich jedoch die Kosten. Der Verbindungssteuermechanismus empfängt Flag-Eingänge, die im ersten Flag-Register F1 31 und im zweiten Flag-Register F2 32 gespeichert werden. Diese Flags können weitergeleitet und zur Änderung der Steuerung verwendet werden. Die Schieberegistermaske 33 sendet die Ausgänge SRM 34 TRUE und SRM 35 COMPLEMENT, die zur Auswahl einer begrenzten Untermenge der SRX 37- und SRY 39-Bitwerte verwendet werden, zur Steuerlogikverbindungsbox 30.

Das X-Register 36 und der SRX-Shifter 37 steuern die logische Verknüpfung und die geometrische Verbindung zu einem benachbarten Verarbeitungselement in X-Dimension. Das Y-Register 38 und das Schieberegister Y SRY 39 dagegen steuern die geometrische und logische Auswahl in Y-Dimension. Gewöhnlich enthalten

das X-Register 36 und das Y-Register 38 die Kartesischen Koordinaten eines Verarbeitungselements im System. Die Shifter SRX 37 und SRY 39 leiten in Verbindung mit SRM 33 benachbarte Bitgruppen von X und Y ab. Die genauen Funktionen von FIG. 3 sind in zwölf Beispielen dargestellt, die zwölf verschiedene Strukturen des polymorphen Maschennetzes beschreiben.

Jedes Verarbeitungselement führt eine arithmetische Logikoperation oder logische Transformationsoperation auf Werten durch, die es erhält, bzw. führt keine Operation, No-op, durch. Zusätzlich zur Durchführung einer Operation oder einer No-op wird das PE mit ausgewählten benachbarten Verarbeitungselementen verbunden. Eine dieser Verbindungen mit einem benachbarten Verarbeitungselement dient als Kurzschlußleitung. Die Kurzschlußverbindung ist im wesentlichen verzögerungsfrei. Die Verbindung findet bei Elektrizitätsgeschwindigkeit (Lichtgeschwindigkeit) statt, und nicht mit einer Verzögerung von einem Zyklus, wie dies bei gewöhnlichen Operationen, einschließlich No-Op, der Fall ist. Damit können mehrere Verarbeitungselemente übergangen werden, um die gewünschte Verbindung mit einem Verarbeitungselement herzustellen, das ein nicht-benachbarter Kartesischer Nachbar ist. Es kann eine Zickzack-Bewegung gemacht werden, um eine Verbindung mit einem entfernten Verarbeitungselement herzustellen, das weder in der gleichen Spalte noch in der gleichen Reihe ist. Obgleich die Feldverbindungen ohne Diagonalen sind, kann auf diese Art und Weise eine Verbindung zu benachbarten diagonalen oder entfernt gelegenen diagonalen oder off-diagonalen entfernten Verarbeitungselementen hergestellt werden.

Obgleich noch weitere Erklärungen folgen werden, ist an dieser Stelle festzuhalten, daß ein polymorphes Maschennetzwerk für die Bildverarbeitung bei entsprechender Programmsteuerung so konfiguriert werden kann, daß es Verarbeitungsschritte ausführt, zu denen die komplexe Zwischenschaltung von Verarbeitungselementen gehört. Jedes Verarbeitungselement führt seine entsprechende arithmetische oder logische Operation,

Transformationsoperation oder No-Op durch, je nachdem, welche Daten es erhalten hat; das Verarbeitungselement kann auch kurzgeschlossen werden, um ohne Zyklusverzögerung übergangen werden zu können.

Es stehen hochentwickelte Verbindungen und Verbindungslogiken als Funktion der Anzahl von Bitwerten in den Strukturregistern und als Funktion der Anzahl von Strukturregistern des Verbindungssteuermechanismus zur Verfügung. Diese hochentwickelten Verbindungen sind jedoch aufgrund der großen Anzahl von Repliken, die von den zahlreichen Verarbeitungselementen benötigt werden, kostspielig. Um die Kosten für die Verarbeitungselemente niedrig zu halten, werden diese nicht nur am Geld, sondern auch an der Komplexität und Pfadlänge gemessen. Das bevorzugte Ausführunsbeispiel weist daher auch ein begrenztes Repertoire von optimierten Verbindungen auf. Dieses Repertoire ist in den FIGUREN 4 bis 20 dargestellt.

FIG. 4 zeigt die Bildung linearer Felder. Es ist ein lineares Feld 41 von West nach Ost, und ein lineares Feld 42 von Nord nach Süd vorhanden. In den FIGUREN 5 und 6 werden zwei Methoden zur Bildung von Reihenbäumen gezeigt. Bei der ersten Methode werden die X-Zeichenketten 45, 46, 47 und 48 bis 51 in FIG. 5 gezeigt. Die X-Zeichenketten der Reihenbäume sind unterschiedlich lang. FIG. 6 zeigt eine herkömmliche Darstellung in Form einer Baumverbindung 52. Bei dieser herkömmlichen Baumkonfiguration filtern 7 Eingänge ein einzelnes Verarbeitungselement in vier Schritten aus.

Die FIGUREN 7-10 zeigen einen Chipbereichsvergleich und eine Kommunikationsentfernungsverbesserung, die sich aus der Verwendung von polymorphen Maschenverarbeitungselementen ergeben haben. Der Chipbereich des polymorphen Maschennetzes 61 in FIG. 7 ist sehr kompakt im Vergleich zur Struktur 62 eines orthogonalen Baums, FIG. 9. Unter Verwendung eines polymorphen Maschennetzes beträgt die durchschnittliche Entfernung in einem 3x3 Fenster (FIG. 8, Struktur 63) 1,5. Das

gleiche 3x3 Fenster 64 in FIG. 10 hat eine durchschnittliche Kommunikationsentfernung von 2,125.

FIG. 11 ist ein Diagramm, das die Bildung von Umkehrreihenbäumen von polymorphen Maschenverarbeitungselementen zeigt. Diese Abbildung ähnelt den in FIG. 5 gezeigten Reihenbäumen. Während ein Reihenbaum Daten von seinen Blättern sammelt, gibt ein Umkehrreihenbaum Daten an seine Blätter ab.

Die FIGUREN 12-20 zeigen ausgewählte Feldkonfigurationen, die durch programmierbare Verbindung polymorpher Maschenverarbeitungselemente einfach zu bilden sind. Einige der Verarbeitungselemente in den Figuren sind als quadratische Kreisanordnung dargestellt, die auf Verarbeitungsoperationen hinweisen. Im Gegensatz zu auftretenden Verarbeitungselementen, die für einfachen Durchlauf verwendet werden und daher als einfache Kreise dargestellt sind. Jedes PE kann je nach Programmierung Durchlauffunktionen und Verarbeitungsoperationen durchführen. Jede Struktur ist auf 16 Bitwerte reduziert und kommt vom entsprechenden Strukturregister.

FIG. 21 ist eine genaue Darstellung des bevorzugten Ausführungsbeispiels eines polymorphen Maschenverarbeitungselements entsprechend der Erfindung. Ein Großteil des gezeigten Mechanismus ist standardmäßig bekannt. Diese Standard-Hardware wird außerhalb des mit unterbrochenen Linien umrahmten Kastens 93 gezeigt. Zu der Standard-Hardware gehört die ALU 96, die die Ausgänge 1 und 2 sowie die entsprechende Eingangsmultiplexe bereitstellt. Die Eingänge kommen vom Befehlsterminal 94 und von den Registern NSE und W 95, die von dem Speicher 1 M1, Speicher 2 M2, dem ALU-Ausgang 1 oder Ausang 2 gespeist werden. Der lokale Speicher bei 101 dient für Berechnungen und Housekeeping im Zusammenhang mit der ALU. Der lokale Speicher 101 kann zum externen Speicher erweitert werden, dessen Inhalt über eine externe Speicherdatenleitung EMD 102 zum Verarbeitungselement geführt wird. Der Inhalt des lokalen und externen Speichers wird vom Multiplexer 103 multiplexiert und mit M1

oder/und M2 verbunden. Die Ausgangssignale out 1 und out 2 werden zum gleichen Verarbeitungselement 93 zurückgeführt sowie zu anderen vom CCM ausgewählten Verarbeitungselementen.

Die EMD-Verbindung und die Busse M1 und M2 sorgen für eine externe Speicherdatenverbindung zu den internen Speichermitteln, Verarbeitungsmitteln und dem Verbindungssteuermechanismus, wobei das Verarbeitungselement gleichzeitig mit dem lokalem Speicher und dem externem Speicher arbeiten kann.

Die Verbindung bei EMD 102 ist wichtig, da sie die direkte Verbindung des einzelenen PE mit einem externen Speicher ermöglicht, der sich im Host H 3 oder in einem selbständigen Speicher befinden kann, der nicht gezeigt wird. Diese EMD-Verbindung, die normalerweise bei einfachen Verarbeitungselementen in einem Feldprozessor nicht vorhanden ist, ermöglicht die Entsprechung von FIG. 3 von einem externen Speicher. Die EMD-Verbindung ermöglicht ebenso ein Ergänzung der Hardware von FIG. 3 sowie eine große Operationsflexibilität und die Einrichtung eines polymorphen Maschennetzwerks der Verarbeitungselemente.

Die in FIG. 3 gezeigte Schalt-Capability ist über die Richtungs- und Logikfunktionssteuerung mit dem externen Speicher und der EMD 102 verbunden. Ein breites Spektrum an Implemtierungsmittel für FIG. 3 ist möglich, angefangen bei sämtlichen in FIG. 3 gezeigten Funktionen, die sich in jedem Verarbeitungselement befinden, bis zu dem Fall, bei dem sich diese Funktionen extern von den PEs befinden, die sich ergebenden Bedingungen jedoch über die Verbindung EMD 102 zum einem Strukturauswahlregister Rp 99 senden.

Die beiden Strukturregister PR0 97 und PR1 98 ermöglichen das verzögerungsfreie Schalten von einer Verbindungsstruktur zur anderen, ohne daß dabei ein Befehlszyklus verloren geht. Das verzögerungsfreie Umschalten wird von einem 1-Bit-Register, dem Strukturauswahlregister (Rp 99) gesteuert. Wenn feststeht,

daß die Werte in einem der beiden Strukturregister nicht mehr länger benötigt werden, kann das Strukturregister mit einer neuen Struktur beladen werden. Das Beladen kann frei sein, d.h. gleichzeitig mit der parallelen ALU-Operation stattfinden. Es darf an dieser Stelle nicht vergessen werden, daß die Verarbeitungselemente in Bildverarbeitungssystemen normalerweise 1-Bit-Verarbeitungselemente und in jedem Fall relativ einfach sind. Das Beladen der Strukturregister 97 und 98, die jeweils 16 Bits aufweisen, ist relativ leistungsintensiv. Das Strukturauswahlregister 99 muß ebenfalls beladen werden, wobei es sich jedoch hierbei um ein 1-Bit-Register handelt.

In manchen Fällen muß die Verarbeitung eingestellt werden, um die Strukturregister 97 und 98 aufzuladen. Wenn dies eintritt, dauert die Aufladung des Strukturregister normalerweise 32 Zyklen, wobei zudem noch ein 33. Zyklus zum Aufladen des Strukturauswahlregisters 99 benötigt wird. In vielen Fällen muß jedoch keine Zeit speziell zum Aufladen der Strukturregister veranschlagt werden. Mit den entsprechenden Befehlen, die als solche bekannt sind, kann der Benutzer ein Bit in das Strukturregister 97 oder ein Bit in das Strukturregister 98 oder ein Bit in das Strukturauswahlregister 99 laden. Während des Zyklus, in dem die ALU 96 eine arithmetische und logische Operation über einen Zeitraum hinweg durchführt, bei dem die Rechen- und Steuereinheit 96 bei voller Kapazität arbeitet, können eines oder alle Register in den CCM geladen werden. Diese Freiladung erfolgt über einen Pfad von einem Befehl in der Menge 94 über die Multiplexer in der ALU 96 und der Rückkoppelung von out 1 oder out 2, vorausgesetzt, das Befehls-Gate 100 ist entsprechend eingestellt, um die Ladefunktion auszuführen.

Die ausgewählten 16 Bitwerte von den Strukturregister 97 und 98 steuern die genaue Einstellung des 4x4 Crossbar-Schalters 104. Siehe hierzu FIG. 2.

Bei einer typischen Operation ist das Verarbeitungselement

entweder kurzgeschlossen oder aktiv. Bei einem Kurzschluß übernimmt ein Strukturregister wie beispielsweise Register 97, das eigens für die Kurzschlußverbindung vorgesehen ist, die Operation und führt eine Kurschlußverbindung über dieses Verarbeitungselement zu einem oder mehreren anderen Verarbeitungselementen durch. Wenn das Verarbeitungselement aktiv ist, ist das Strukturauswahlregister 99 aktiv und schaltet die Steuerung vom Standardstrukturregister 97, das für das Übergehen infolge des Kurzschlusses sorgt, auf das andere Strukturregister 98 um, das die Ein- und Ausgänge für die entsprechende Aktivität leitet.

Diese Aktivitäten werden in den nachfolgenden Abschnitten näher erläutert.

Das in den FIGUREN 1-3 gezeigte polymorphe Maschennetz verfügt über eine Anzahl von Strukturen, die durch die Komplexität seines Verbindungssteuermechanismus CCM begrenzt werden. Das CCM-Strukturrepertoire, das die Verbindung dieser Strukturen darstellt, eignet sich optimal für die Verbindung von Berechnungsarten, die separat ausgeführt werden.

Für jede Struktur ist in der Erfindung ein Steueralgorithmus beschrieben. Sämtliche Steueralgorithmen sind einfach zu implementieren und verwenden vor allem die gleiche Hardware, um die gewünschte Struktur on-line zu erzeugen.

In der Erfindung wird ein Hardware-Mechanismus aufgezeigt, der die Bildung aller Strukturen auf systematische und einheitliche Weise durchführt. Der Mechanismus ist einfach zu implementieren und eignet sich vor allem für VLSI-Anwendungen.

Eine hervorstechende Eigenschaft des polymorphen Maschennetzes besteht darin, daß zahlreiche Algorithmen linearer Komplexität (O(N)) auf eine logarithmische Komplexität (O(log N), ein theoretisches Optimal, reduziert werden. Die N/logN Beschleunigung wird aufgrund der neuen Architektur erzielt; für ein

Netzwerk von 1024x1024 beträgt die Beschleunigung 100.

Bei der Computer Vision wird das sich nach der ikonischen Verarbeitung durch eine Struktur (Masche) ergebende Bild nicht außerhalb des Netzwerks gebracht. Es wird dagegegen noch einmal von einer anderen Struktur (z.B. Baum) verarbeitet, um die ikonischen Daten in symbolische Daten umzuwandeln (z.B. wie viele Pixel sind vorhanden, deren Werte größer als 133 sind?). Aufgrund der polymorphen Capability müssen die Daten nicht ausgegeben werden, und daher wird die I/O-Rate bedeutend reduziert (z.B. um fünf Größenordnungen für das Beispiel "Wie viele" in einem 1024x1024 Bild). Diese I/O-Reduzierung trägt zur Beschleunigung aufgrund einer zusammengesetzten Berechnung bei.

Eine andere Struktur, ein Diagonal-Span-Baum, kann vom polymorphen Maschennetz gebildet werden, um die Berechnung von Ax+By+C in Logarithmuszeit zu erleichtern, wobei A, B und C konstant sind und (x,y) die Koordinate eines Pixel ist. Diese Capability ist vor allem bei der Computergraphik und Computer Vision nützlich. Bei der Computergraphik kann es vor allem für die Anzeige konvexer Polygone, das Erzeugen von Schatten, Clipping, das Zeichnen von Kugeln, das Berechnen von adaptiver Histrogrammgleichstellung, der Texturabbildung und Antialiasing verwendet werden. Bei der Computer Vision ist diese Funktion für die Erzeugung von Zeilenmasken, Bandmasken und Polygonmasken nützlich. Darüber hinaus kann es für die Berechnung von Fast Hough Transform und seine Umkehrung zur Zeilendetektion in Rauschbildern und anderen Anwendungen eingesetzt werden.

Das bevorzugte Ausführungsbeispiel beinhaltet einen Hardware-Mechanismus, der zwölf nützliche zelluläre Automatenstrukturen sowie zwölf Steueralgorithmen erzeugt, einen für jede Struktur, um das polymorphe Maschennetz unter Software-Steuerung in die entsprechende Struktur umzuwandeln. In den folgenden Abschnitten werden der polymorphe Hardware-Mechanis-

mus und die Steueralgorithmen beschrieben.

Eine wichtige Eigenschaft des polymorphen Maschennetzes ist die Fähigkeit, sich selbst adaptiv an die Verarbeitungsbedingungen "anzupassen". Wie oben bereits beschrieben können die Strukturregister 97 und 98 parallel zur ALU-Operation beladen werden. Daraus ergibt sich, daß jedes PE P-Strukturen (P1...Pp) zur Verfügung hat.

Das Umbilden, das adaptiv entsprechend der der Bedingung C der Verarbeitung erfolgt, ermöglicht jeder PE eine Struktur Pi als eine Funktion C vorauszusetzen. Anfänglich starten alle PEs von einer Struktur aus, angenommen Pi, und die Verarbeitung beginnt unter der anfänglichen Struktur. Bei fortschreitender Verarbeitung erfaßt jedes PE seine lokale Bedingung (dies kann beispielsweise eine Konvergenzprüfung des Wertes sein) und legt fest, ob es in der aktuellen Struktur bleibt, oder diese durch eine neue Struktur ersetzt.

Eine Bedingung kann global sein, d.h., daß der Host alle PE-Bedingungen zusammen erfassen kann und dann die Strukturwahl festlegt und an alle PEs rückkoppelt. Eine Bedingung kann auch lokal sein. Die neue Struktur kann zu einem einzelnen PE gesendet werden, oder je nach Bedarf zu einer Gruppe von PEs.

Die Auswahl einer neuen Struktur kann auf verschiedenen Wegen erfolgen.

(1) vorgeplant:

eine Strukturfolge (z.B. P1 P2 ... Pp) ist geplant und wird in dieser Reihenfolge angenommen. Wenn der Bedarf nach einer neuen Struktur erfaßt wird (während P1 verwendet wird), wird die nächste Struktur P(i + 1) verwendet. Danach wird P(i + 2) parallel mit der Operation in das unbenutzte Strukturregister geladen, sofern dies möglich ist.

(2) Funktion von C:

Ein Beispiel für die adaptive Umbildung sieht folgendermaßen aus:

Eine Struktur eines 3x3 Fensters wird als Pi eingesetzt, um eine Filteroperation durchzuführen, und ein Index wird als Bedingung C gekennzeichnet, um die Effektivität der Filterung zu messen. Dabei soll die Fenstergröße erhöht werden, wenn C nicht zufriedenstellend ist. In dieser Hinsicht wird P2 als 5x5 Fenster, P3 als 7x7, P4 als 9x9, usw. eingesetzt.

Ein weiteres Beispiel für die adaptive Umbildung ist die "Soft Fail-Anwendung". Gewöhnlich wird in jedem PE eine Bedingung C definiert, so daß eine Störung angezeigt wird. Diese Bedingung kann eingesetzt werden, um eine neue geeignete Verbindungsstruktur Pi festzulegen.

POLYMORPHER MASCHENMECHANISMUS

Das polymorphe Maschennetz (Figur 1) besteht aus einem Feld 1 von MxM Verarbeitungselementen (PE). Jedes PE verfügt über vier physische Leitungen, die mit vier nicht-diagonalen Nachbarn kommunizieren, d.h. eine Leitung für jeden Nachbarn. Diese Kommunikationsleitungen werden im Fall von PE 4 als E, W, S und N bezeichnet. PE 5 wird vergrößert gezeigt, um die Steuer- und Recheneinheit (ALU) 6, den Speicher (M) 7 und den Verbindungssteuermechanismus (CCM) 8 darzustellen, wobei die ALU 6 und der Speicher 7 standardmäßige Einrichtungen bei Verarbeitungselementen von Bildprozessoren sind. Der Verbindungssteuermechnismus ist dagegen nicht standardmäßig vorhanden, im Fall der Erfindung jedoch konfigurationsspezifisch.

Aus den FIGUREN 1 und 2 geht hervor, daß jedes PE aus drei Funktionsblöcken besteht: dem Verbindungssteuermechanismus 8, dem Speicherblock und dem Steuer- und Recheneinheitsblock

(ALU-Block) 6. Der Verbindungssteuermechanismus 8 verwendet die vier Leitungen (E, W, N und S), den ALU-Ausgang und den Speicher (M) als Eingänge und leitet sie als Ausgänge zurück. Das Weiterleiten erfolgt durch "SHORT-CIRCUITing" (Kurzschließen) eines Eingangs A (beispielsweise 10 in FIG. 2) zu einem Ausgang B. Das auf Leitung A erscheinende Signal ist logisch das von Leitung B, wobei A und B beliebige Eingänge zur Verbindungssteuereinheit sein können. "SHORT-WE" beispielsweise stellt Leitung W 24 und Leitung E 26 logisch gleich.

FIG. 3 zeigt die Funktionseinheiten von CCM 8. Die Maßnahme "SHORT-CIRCUIT" ist bedingt und die Bedingungen wurden von dem in Figur 3 gezeigten Verbindungssteuermechanismus aufgestellt. Jedes PE hat zwei 1-Bit-Flags F1 31 und F2 32 zur Erzeugung der Bedingungssignale. Jedes PE verfügt über ein Schieberegister SRM 33, das sowohl in logischer als auch in arithmetischer Richtung verschieben kann und die wahren und komplementären Ausgänge 34 und 35 bereitstellt.

Jedes PE ist mit einem Registerpaar X 36 und Y 37 ausgestattet, wobei X die Reihenposition der PEs ($0 \leq X \leq M-1$) und Y die Spaltenposition der PEs ($0 \leq Y \leq M-1$) enthält. Die beiden Register X und Y haben jeweils ein Schieberegister SRX 38 und SRY 39, in die der Inhalt von X bzw. Y geladen werden kann und die in logischer und arithmetischer Richtung verschoben werden können. Das aus SRX verschobene Bit ist BSRX und das aus SRY verschobene Bit ist BSRY.

Es sind mehrere Funktionen vorhanden, um die Bedingung herzustellen, auf der die Maßnahme "SHORT_CIRCUIT" beruht.

LOAD reg value: mit dieser Funktion wird der "Wert" in SRM, F1 oder F2 durch einen Befehl oder über einen Speicher geladen;

COPYSR reg: mit dieser Funktion wird Reg X auf Reg SRX oder Y auf SRY oder beide kopiert.

AND/OR/XOR reg: mit dieser Funktion werden folgende Schritte durchgeführt:

"AND/OR/XOR" X mit SRX und X mit SRM (oder umgekehrtem SRM);
"AND/OR/XOR" Y mit SRY und Y mit SRM (oder umgekehrtem SRM);
beide oben genannten Schritte;

ANDALLBIT reg: mit dieser Funktion wird eine "AND-Operation" auf allen Bits von Reg durchgeführt; es wird dabei ein Bedingungsbit XANDALL erzeugt, wenn Reg gleich X ist, oder eine Bedingung YANDALL, wenn Reg gleich Y ist, oder beide Bedingungsbits.

Die Maßnahme "SHORT_CIRCUIT" beruht dann auf der Verbindung von BSRX, BSRY, XANDALL, YANDALL, F1 und F2.

Die beiden restlichen Funktionsblöcke des polymorphen Maschennetzes entsprechen dem herkömmlichen Design. Der Speicherblock ist ein Speicher, der ein Bit zum Verbindungssteuerblock sendet und/oder ein Bit pro Maschinenzyklus vom Verbindungssteuerblock annimmt. Obgleich die ALU dem herkömmlichen Design entspricht, wählt auf die "bedingte" Antwort auf die Bitverbindung BSRX, BSRY, XANDALL und YANDALL eine "SEND-" oder "RECEIVE-Maßnahme" zusammen mit der "SHORT_CIRCUIT-Maßnahme".

Im Zusammenhang mit dem polymorphen Maschenmechanismus werden nachfolgend zwölf Strukturen beschrieben, die vom polymorphen Maschennetz gebildet werden. Die entsprechenden Steueralgorithmen werden der Reihenfolge nach beschrieben.

STEUERALGORITHMEN

(P1) Lineares Feld

Aus Figur 4 geht hervor, daß ein lineares Reihenfeld von MxM Länge und ein lineares Spaltenfeld der gleichen Länge aus einem MxM polymorphen Maschennetz gebildet werden kann, indem

S von PE (M-1,i) mit N von PE (0,i+1) und E von PE (i, M-1) mit W von PE (i+1,0) verbunden wird. N von PE (0,0) und S von PE (M-1, M-1) bilden den Anfang bzw. das Ende des linearen Spaltenfeldes, während W von PE (0,0) und E von PE (M-1, M-1) den Anfang bzw. das Ende des linearen Reihenfeldes bilden.

Der Steueralgorithmus sieht folgendermaßen aus: Pro PE-Zylus,

```
LINEAR ()

{

     MEM = W;/*Maßnahme 1*/
     E = MEM;/*Maßnahme 2*/
     MEM = N;/*Maßnahme 3*/
     S = MEM;/*Maßnahme 4*/

}
```

Maßnahme (1) setzt die Dateneinheit auf W in MEM am Ende des Zyklus, während Maßnahme (2) den Inhalt von MEM (am Anfang des Zyklus) auf E setzt. Zusammen erzeugen die Maßnahmen (1) und (2) das Reihenfeld. Die durch W von PE (0,0) eingespeisten Daten verlaufen in östlicher Richtung, und nach M Zyklen ist die erste Reihe gefüllt. Nach weiteren M Zyklen gehen die Daten in der ersten Reihe in die zweite Reihe, während die neuen Daten in die erste Reihe eingefügt werden. Die Maßnahmen (3) und (4) sorgen auf ähnliche Art und Weise für ein lineares Spaltenfeld.

Die Bildung des linearen Feldes ist nicht bedingt. Alle PEs führen die gleichen Maßnahmen durch. Der Mechanismus im Verbindungssteuermechanismus wird nicht verwendet.

(P2) Reihenbaum

Mit Hilfe des folgenden Steueralgorithmus können M Bäume

(einer pro Reihe) von dem polymorphen Maschennetz gebildet werden.

```
ROW-TREE ()

{

int t;/*t ist der Zeitschritt*/
int pid0;/*Spaltenposition eines PE, ID0 verarbeiten*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int treemask=1;/*ein Flag zur Bildung des Baums*/

     for(t=0;t<logM;t++){
          if(¬treemask)
               {SHORT_WE;DISABLE;}
          if(treemask && ¬pid0<t>)
               E = MEM;
          if(treemask && pid0<1>)
               MEM = W;
          treemask = treemask & pid0<t>;
     }
}
```

Figur 5 zeigt den oben dargestellten Steueralgorithmus für den Fall eines 8-PE. Bei t=0 sind die "treemasks" (Baummasken) für alle PEs 1, daher wird jedes PE aktiviert und die "geraden PEs" senden Daten zu den "ungeraden PEs". Dies wird durch einen Pfeil zwischen jedem Paar von "ungeraden/geraden" PEs angezeigt. Dadurch wird auch die unterste Stufe des Baums gebildet.

Bei t=1 werden durch Untersuchung der Baummaske nur die PEs mit pid0<0>=1 (niedrigstes Bit von pid0) aktiviert. Die nicht aktivierten PEs werden nicht im Kreis gezeigt, sie stellen jedoch die Verbindung zwischen PE 1 und 3 sowie PE 5 und 7 durch die Maßnahme SHORT_WE her. Dadurch wird die zweite Stufe

des Baums gebildet.

Die höchste Stufe des Baums wird durch die Steuerung bei t=2 gebildet. An dieser Stelle werden nur PE 3 und PE 7 aktiviert; die restlichen PEs stellen die Verbindung durch "SHORT" des W-E-Pfades her, führen jedoch keine Operation durch (oder ändern nicht den Status von MEM).

Eine andere Darstellung dieses Steueralgorithmus erscheint in Figur 6, wo die Knoten von jeder Baumstufe von der Prozessoridentifikation (pid) der PEs markiert werden.

Die Baumstruktur ist vor allem für ein Teile-und-Herrsche-Paradigma nützlich. Dedizierte Baummaschinen wurden für spezielle Berechnungen gebaut. Die Komplexität der Algorithmen in diesem Paradigma ist gewöhnlich O(log N), wobei N die Größe der Eingangsdaten ist. Der gleiche Algorithmus braucht O(N) Ausführungszeit in einem Maschennetz; dies ist eine 1024:10 Beschleunigung für N=1024. Zu den wichtige Algorithmen in dieser Kategorie gehören MAX, MIN, k-te größter, mittlerer, einige/keine usw.

Wenn der Mechanismus im Verbindungssteuerblock verwendet wird, benutzt der Steueralgorithmus das Register X für pid0, das Register SRX und das Flag F1 für die "Baummaske". Der Inhalt des X-Registers wird in SRX kopiert, so daß das pid<t>-Bit bei Zeitschritt 1 in BSRX verschoben und dann in einer AND-Operation mit der "Baummaske" verbunden wird, um die Endbedingung abzuleiten.

(P3) Spaltenbaum

Ähnlich wie bei Reihenbäumen können M Spaltenbäume vom polymorphen Maschennetz gebildet werden, wobei die Reihenposition der PEs, pid 1, die Steuerung ist, und N-S Pfad die Verbindung. Der Steueralgorithmus sieht folgendermaßen aus:

```
COLUMN-TREE ()

{
int t;/*t ist der Zeitschritt*/
int pid1;/*Reihenposition eines PE*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int treemask=1;/*ein Flag zur Bildung des Baums*/

     for(t=0;t<logM;t++){
          if(¬treemask)
               {SHORT_NS;DISABLE;}
          if(treemask && ¬pid1<t>)
               S = MEM;
          if(treemask && pid0<t>)
               MEM = N;
          treemask = treemask & pid0<t>;
     }
}
```

Spaltenbäume können verwendet werden, um Daten, die spaltenweise im Maschennetz verteilt sind, zu übertragen und Algorithmen mit linearer Komplexität (O(N)) wie im Abschnitt mit den Reihenbäumen beschrieben in Algorithmen mit logarithmischer Komplexität (O(log N)) umzuwandeln.

Ähnlich wie der ROW_TREE-Steueralgorithmus verwendet der COLUMN_TREE-Steueralgorithmus das Register Y für pid1, SRY zum Kopieren von Y und F2 für die "Baummaske". Die Bedingung für SHORT_NS beruht auf einer AND-Operation von F2 und BSRY, aus der pid1<t> beim Zeitschritt t hervorgeht.

(P4) Orthogonaler Baum

Ein orthogonaler Baum (FIG. 9) ist ein nützliches Netzwerk zum Sortieren, für Matrixoperationen, minimal spannender Baum, FFT und andere Graphalgorithmen. Der Baum kann vom polymorphen

Maschennetz mit Hilfe des untenstehenden ORTH_TREE-Steueralgorithmus durch Verbinden der Reihen- und Spaltenbäume gebildet werden.

```
ORTH_TREE ()

{
int t;/*t ist der Zeitschritt*/
int pid0;/*Spaltenposition eines PE, ID0 verarbeiten*/
int pid1;/*Reihenposition eines PE*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int hmask=1, vmask=1;/*Flags zur Bildung des Baums*/

     for(t=0;t<logM,t++){
     /*Zyklus 1*/
          if(¬hmask)
               {SHORT_WE;DISABLE;}
          if(hmask && ¬pid0<t>)
               E = MEM;
          if(hmask && pid0<t>)
               MEM = W;
          hmask = hmask & pid0<t>;

     /*Zyklus 2*/
          if(¬vmask)
               {SHORT_NS;DISABLE;}
          if(vmask && ¬pid1<t>)
               S = MEM;
          if(vmask && pid1<t>)
               MEM = N;
          vmask = vmask & pid1<t>;
     }
}
```

Die Hauptvorteile eines orthogonalen Baums von einem polymorphen Maschennetz liegen (1) bei der Reduzierung des Chipbe-

reichs: der Chipbereich, der für das Maschennetz und den orthogonalen Baum benötigt wird, ist O(N**2) bzw. O((N**2)*(log N)**2), wobei N die Seitengröße des Maschennetzes und die Anzahl der Blätter des orthogonalen Baums darstellt. Dadurch wird eine Einsparung vom Faktor (log N)**2 erzielt. Bei N=1024 beträgt der vom polymorphen Maschennetz beanspruchte Chipbereich 1/100 des orthogonalen Baums.

(2) effiziente Nachbarschaftsoperationen: PEs in orthogonalen Bäumen sind wegen der Bildverarbeitung nicht mit ihren geographisch nächstgelegenen Nachbarn verbunden. Viele wichtige Nachbarschaftsoperationen können nicht effizient durchgeführt werden, weil keine direkten Kommunikationspfade vorhanden sind. Mehr als die Hälfte aller Daten in einem 3x3 Fenster müssen eine Stufe nach oben im Baum gebracht werden, bevor sie zur Mitte des Fensters gehen. Die durchschnittliche Entfernung zwischen Daten in einem 3x3 Fenster beträgt 2,125 gegenüber 1,5 in einem polymorphen Maschennetz. In den Figuren 8 und 10 ist ein Beispiel eines 3x3 Fensters für den orthogonalen Baum dargestellt, wobei die Zahl im Kreis die Entfernung zwischen der Dateneinheit und der Fenstermitte angibt.

Wie in Figur 7 für N=4 zusammengefaßt ist, beträgt das Verhältnis des Chipbereichs zwischen dem polymorphen Maschennetz und dem orthogonalen Baum 16:46 und das Verhältnis der durchschnittlichen Entfernung in einem 3x3 Fenster 1,5:2,1 (Figur 8 und 10).

Der Steueralgorithmus ORTH_TREE verwendet das Register X für pid0, SRX zur Kopie von pid0, F1 für hmask und erzeugt pid0<t> beim Zeitschritt t in BSRX. Der zweite Steueralgorithmus verwendet das Register Y für pid1, SRY zur Kopie von pid1, F2 für vmask und erzeugt pid1<t> beim Zeitschritt t in BSRY. Die bedingten SHORT_WE und SHORT_NS beruhen auf BSRX, BSRY, F1 und F2.

(P5) Umkehrreihenbaum

Der RR-Baum ist ein Top-Down-Baum (im Gegensatz zu Bäumen, die Bottom-Up sind), der von der oberen Ebene zur unteren Ebene des Baums gebildet werden kann, d.h. ein Umkehrprozeß eines Reihenbaums. Der Steueralgorithmus ist nachfolgend dargestellt.

```
RR-TREE ()

{
int t;/*t ist der Zeitschritt*/
int pid0;/*Spaltenposition eines PE*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int treemask=M%2;/*ein Flag zur Bildung des Baums*/
int mask;/*eine Zwischenbedingung*/

     for (t=0;t<logM;t++){
          mask = ANDALLBIT (¬treemask | pid0);
          if(¬mask)
               {SHORT_WE;DISABLE;}
          if(mask && pid0<logM-t-1>)
               W = MEM;
          if(mask && ¬pid0<logM-t-1>)
               MEM = E;
          treemask = ASHIFT (treemask,1);
     }
}
```

Anhand eines 8-PE-Beispiels in Figur 11 kann der Steueralgorithmus wie folgt beschrieben werden. Die Flag-Baummaske wurde als eine Hälfte der Gesamtanzahl von PEs initialisiert (z.B. 4 = 100). Die UMGEKEHRTE "Baummaske" wird mit pid0 einer OR-Operation unterzogen und das Ergebnis wird zu ANDALLBIT weitergeleitet, das eine '1' in die 'Maske' bringt, wenn alle Bits des Ergebnisses '1' sind, andernfalls wird eine '0' gebracht. Die PEs in der Maske=1 (z.B. PE 3 und 7) sind Teil des Baums. Der Rest, der keinen Baumknoten darstellt, deakti-

viert sich selbst und schließt (SHORT) seinen W-E-Pfad kurz, um die Baumverbindung herzustellen. Bei PE 3 und 7 wird das Bit 2 von pid0 weiter geprüft, eine '1' in diesem Bit veranlaßt PE 7 die Dateneinheit zum Empfänger PE 3 zu schicken, dessen Bit 2 '0' ist. Dadurch wird die obere Ebene des Baums bei t=0 gebildet.

Am Ende von t=0 wird die Baummaske arithmetisch um ein Bit nach rechts verschoben. Es wird zu 110 für den nächsten Zeitschritt.

Bei t=1 stellt der gleiche Prozeß fest, daß PE 1, 3, 5 und 7 Baumknoten sind, und daß PE 7 Daten zu PE 5 und PE 3 Daten zu PE 1 sendet. Am Ende von t=1 wird die Baummaske zu 111.

Bei t=2 ist jedes PE ein Baumknoten und jedes PE mit ungeradem pid0 sendet Daten zu seinem Nachbarn mit niedrigerem geraden pid0.

Unter Verwendung des Mechanismus im Verbindungssteuerblock wird die "Baummaske" in SRM geladen und X auf SRX kopiert. Das Register X wird zuerst einer "OR-Operation" mit dem UMGEKEHRTEN SRM unterzogen; das Ergebnis wird einer "ANDALLBIT-Operation" unterzogen, um die "Maske" zu erzeugen. SRX wird logisch nach links verschoben, um pid0<logM-t-1> in BSRX beim Zeitschritt t zu erzeugen. Dies wird dazu verwendet, die SEND/RECEIVE-Maßnahme für ein Paar von Baumknoten zu steuern.

Der RR-Baum wird hauptsächlich zur Weiterleitung einer Dateneinheit zu allen Baumknoten verwendet, die je nach Position im Baum unterschiedliche Operationen auf dieser Dateneinheit durchführen. Für Anwendungen im Computergraphikbereich ist diese Struktur besonders nützlich, da damit jedes PE A*X gleichzeitig erzeugen kann, wobei A eine Konstante und X pid0 ist. Die parallele Berechnung von A*X ermöglicht die schnelle Erzeugung einer Zeile. Darauf wird noch näher bei

einer anderen Struktur, dem sogenannten Diagonal-Span-Tree (P12) eingegangen.

Allgemein wird ein Umkehrbaum dazu verwendet, eine symbolische Darstellung in einem Parameterbereich in eine ikonische Darstellung in einem Bildbereich umzuwandeln. Der Algorithmus wird dann ikonisch in einer Parallelität ausgeführt, die im polymorphen Maschennetz zur Verfügung steht.

Obgleich der Steueralgorithmus das PE mit dem höchsten pid0 als Wurzel des Baums verwendet, kann auch das PE mit dem niedrigsten pid0 als Wurzel verwendet werden, wobei der Steueralgorithmus die gleiche Komplexität aufweist.

(P6) Umkehrspaltenbäume (RC-Baum)

Ähnlich wie beim RR-Baum kann der RC-Baum mit Hilfe von pid1 als Steuerung und N-S als Pfad zur Herstellung einer Baumverbindung verwendet werden. Dies geht aus dem folgendem Steueralgorithmus hervor.

```
RC-TREE ()

{
int t;/*t ist der Zeitschritt*/
int pid1;/*Reihenposition eines PE*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int treemask=M%2;/*ein Flag zur Bildung des Baums*/
int mask;/*eine Zwischenbedingung*/

     for(t=0;t<logM;t++){
          mask = ANDALLBIT (¬treemask | pid1);
          if(¬mask)
               {SHORT_NS;DISABLE;}
          if(mask && pid1<logM-t-1>)
               N = MEM;
```

```
          if(mask && ¬pid1<logM-t-1>)
               MEM = S;
          treemask = ASHIFT (treemask,1);
     }
}
```

Die Eigenschaften der RC-Bäume sind die gleichen wie bei den RR-Bäumen, mit Ausnahme davon, daß sich die RC-Bäume auf die Daten in der Spalte des Maschennetzes beziehen.

(P7) Reihenbus

Für Rundspruchzwecke ist vor allem ein Bus eine sehr nützliche Struktur, dessen Rundspruchentfernung die kleinste ist. Für jede Reihe des polymorphen Maschennetzes kann mit Hilfe des folgenden Steueralgorithmus ein Bus gebildet werden.

```
ROW_BUS ()

{
int sender;/*ID für den Sender*/
int pid0;

          SHORT_WE;
          if(pid0 == sender)
          E = MEM;
     else
          MEM = W;
}
```

Ein PE in der Reihe wird als "Sender" bezeichnet, während die übrigen PEs Empfänger sind. Alle PEs schließen ("SHORT") ihren E-W-Pfad kurz, um den Bus herzustellen. Der Sender sendet die Daten zu E (oder W), während die Empfänger die Daten von W (oder E) empfangen können. (Wenn in einem anderen Fall eine Dateneinheit von einem externen Controller zu W oder E geführt wird, ist kein "Sender" vorhanden und alle PEs sind

"Empfänger").

Unter Verwendung des Mechanismus im Verbindungssteuerblock wird der "Sender" in SRM geladen. Das "UMGEKEHRTE" SRM wird in einer "XOR-Operation" mit Register X verbunden, das pid0 speichert. Die sich ergebenden Bits werden einer "ANDALLBIT-Operation" unterzogen. Eine '1' in XANDALL legt das PE als Sender fest; die PEs mit XANDALL=0 sind die Empfänger.

(P8) Spaltenbus

Ähnlich wie beim Reihenbus kann für jede Spalte des polymorphen Maschennetzes ein Spaltenbus gebildet werden, indem entsprechend dem folgenden Steueralgorithmus pid1 als Steuerung und N-S als Pfad verwendet wird.

```
COLUMN_BUS ()

{

int sender;/*ID für den Sender*/
int pid1;

     SHORT_NS;
     if(pid1 = = sender)
          S = MEM;
     else
          MEM = N;
}
```

Die Eigenschaften des Spaltenbusses sind die gleichen wie beim Reihenbus.

Zusammen können der Reihenbus und der Spaltenbus verwendet werden, um eine gemeinsame Dateneinheit an alle PEs in einem Maschennetz in zwei Schritten weiterzuleiten. In einem ersten Schritt kann die allgemeine Dateneinheit an alle PEs in der

oberen Reihe weitergeleitet werden; im zweiten Schritt kann das PE in der oberen Reihe die allgemeine Dateneinheit an alle anderen PEs in Spaltenrichtung weiterleiten.

(P9) Pyramide

Eine Pyramidenkonfiguration eignet sich vor allem bei der Bildverarbeitung und Computer Vision, da damit Mehrfachauflösungsbilder verarbeiten werden können. Diese Struktur kann mittels des folgenden Steueralgorithmus vom Maschennetz gebildet werden:

```
PYRAMID()

{
int t;/*t ist der Zeitschritt*/
int pid0;/*Spaltenposition eines PE*/
int pid1;/*Reihenposition*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int hmask=1, vmask=1;/*2 Flags zur Bildung der Pyramide*/

     for(t=0;t<logM;t++){
     /*Zyklus 1 Maßnahme*/
          if(¬mask | ¬vmask)
               {SHORT_WE;SHORT_NS;DISABLE;}
          if(hmask && vmask && ¬pid0<t> && ¬pid1<t>)
               E = MEM;
          if(hmask && vmask && pid0<t> && ¬pid1<t>)
               {N = MEM; MEM1 = W;}
          if(hmask && vmask && ¬pid0<t> && pid1<t>)
               E = MEM;
          if(hmask && vmask && pid0<t> && pid1<t>)
               {MEM0 = N; MEM2 = W;}
     /*Zyklus 2 Maßnahme*/
          if(¬mask | ¬vmask)
               {SHORT_WE;SHORT_NS;DISABLE;}
```

```
            if(hmask && vmask && ¬pid0<t> && ¬pid<t>)
                 NO_ACTION;
            if(hmask && vmask && pid0<t> && ¬pid1<t>)
                 S = MEM1;
            if(hmask && vmask && ¬pid0<t> && pid1<t>)
                 NO_ACTION;
            if(hmask && vmask && pid0<t> && pid1<t>)
                 MEM1 = N;

            hmask = hmask & pid0<t>;
            vmask = vmask & pid1<t>;
       }
}
```

Der Steueralgorithmus besteht aus log M Schritten und in jedem Schritt sind zwei Steuerzyklen. Mit anderen Worten heißt dies, daß jeder Schritt eine Stufe der Pyramide in zwei PE-Zyklen bildet.

Die Figuren 12, 13 und 14 zeigen einen Steueralgorithmus für eine Pyramide in einem 8x8 Maschennetz. Die beiden Masken hmask (für Reihe) und vmask (für Spalte) werden als '1' initialisiert, so daß alle PEs im Maschennetz beim ersten Zeitschritt "aktiviert" werden. Bei t=0 sind alle PEs aktiv und jeweils 2x2 PEs bilden eine Gruppe. Diese vier (2x2) PEs sind die NW-, NE-, SW- und SE-Söhne der Pyramide und der Vater ist der gleiche wie der SE-Sohn. Die Aktivitäten der vier Söhne werden durch die Bits von pid0<1> und pid1<1> unterschieden. Der SE-Sohn (oder der Vater), der als pid0<0>=pid1<0>=1 gekennzeichnet ist, empfängt Daten von den Söhnen SW (pid0<0>=0 und pid1<0>=1) und NE (pid0<0>=1 und pid1<0>=0) im ersten Zyklus. In diesem Zyklus leitet der NW-Sohn seine Daten zum NE-Sohn weiter; diese Daten werden dann vom Vater im zweiten Zyklus empfangen. Im zweiten Zyklus sind nur der NE-Sohn und der Vater vom Senden und Empfangen betroffen; die anderen beiden PEs sind inaktiv. Sowohl vmask und hmask werden aktualisiert, um die Verbindung des nächsten

Zeitschritts zu steuern.

Bei t=1 bilden vier PEs vier Söhne und einen Vater für die nächste Stufe der Pyramide. Diese vier PEs befinden sich wie in Figur 13 gezeigt in einem 4x4 Maschennetz. Die Aktivität der vier Söhne und des Vaters ist die gleiche wie bei t=0, mit der Ausnahme, daß die PEs in den geraden Reihen oder Spalten deaktiviert sind. Diese deaktivierten "nicht in der Pyramide befindlichen" PEs schließen (SHORT) ihr W zur E-Leitung und N zur S-Leitung kurz, um die Pyramidenverbindung herzustellen.

Die PEs, die die letzte Stufe der Pyramide bilden, werden in Figur 14 gezeigt. Ihre Aktivitäten sind die gleichen wie bei den beiden vorangegangenen Schritten.

Orthogonal zu der oben beschriebenen Pyramidenstruktur hat die Pyramidenstruktur eine Maschenverbindung auf jeder Stufe. Für jeden Knoten in der Pyramide sind vier Nachbarn (N, S, E, W) auf der gleichen Stufe vorhanden, im Gegensatz zu vier Söhnen auf der Stufe darunter und einem Vater auf der Stufe darüber. Dieses Verhältnis wird in Figur 15 gezeigt.

Der Steueralgorithmus für die Nachbarn auf der gleichen Stufe ist im Steueralgorithmus der oben beschriebenen Pyramide integriert. Bei t=0 (Figur 8a) werden die Nachbarn der gleichen Stufe beispielsweise vom ursprünglichen Maschennetz verbunden. Bei t=1 sind die Nachbarn der gleichen Stufe in jeder zweiten Reihe und Spalte verteilt, und die Maschenverbindung wurde durch den oben erwähnten Steueralgorithmus PYRAMID hergestellt.

Um den Inhalt der Nachbarn auf der gleichen Stufe der Pyramide zu erhalten, werden zwei Steuerzyklen zu jedem Schritt des Steueralgorithmus PYRAMID wie folgt hinzugefügt. In Zyklus 3 ist der Inhalt von N vorhanden, und in Zyklus 4 der Inhalt von S und E.

```
PYRAMID ()

{
int t;/*t ist der Zeitschritt*/
int pid0:/*Spaltenposition eines PE*/
int pid1:/*Reihenposition*/
int M, logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int hmask=1, vmask=1;/*2 Flags zur Bildung der Pyramide*/

     for(t=0;t<logM;t++){
     /*Zyklus 1 Maßnahme*/
          if(¬hmask | ¬vmask)
               {SHORT_WE;SHORT_NS;DISABLE;}
          if(hmask && vmask && ¬pid0<t> && ¬pid1<t>)
               E = MEM;
          if(hmask && vmask && pid0<t> && ¬pid1<t>)
               {N = MEM; MEM1 = W;}
          if(hmask && vmask && ¬pid0<t> && pid1<t>)
               E = MEM;
          if(hmask && vmask && pid0<t> && pid1<t>)
               {MEM0 = N; MEM2 = W;}

     /*Zyklus 2 Maßnahme*/
          if(¬hmask | ¬vmask)
               {SHORT_WE;SHORT_NS;DISABLE;}
          if(hmask && vmask && ¬pid0<t> && ¬pid1<t>)
               NO_ACTION;
          if(hmask && vmask && pid0<t> && ¬pid1<t>)
               S = MEM1;
          if(hmask && vmask && ¬pid0<t> && pid1<t>)
               NO_ACTION;
          if(hmask && vmask && pid0<t> && pid1<t>)
               MEM1 = N;

     /*Zyklus 3 Maßnahme*/
          if(¬hmask | ¬vmask)
```

```
                {SHORT_WE;SHORT_NS;DISABLE;}
          if(hmask && vmask){
                S = MEM3;
                E = MEM4;
                MEM3 = N;
                MEM4 = W;
          }

     /*Zyklus 4 Maßnahme*/
          if(¬hmask | ¬vmask)
                {SHORT_WE;SHORT_NS;DISABLE;}
          if(hmask && vmask){
                N = MEM5;
                W = MEM6;
                MEM5 = S;
                MEM6 = E;
          }

          hmask = hmask & pid0<t>;
          vmask = vmask & pid1<t>;
     }
}
```

Unter Verwendung des Mechanismus im Verbindungssteuerblock werden hmask und vmask in F1 bzw. F2 geladen. pid0 in Register X und pid1 in Register Y werden in SRX bzw. SRY kopiert. Bei der logischen Verschiebung nach rechts enthalten BSRX und BSRY pid0<t> und pid1<t> bei Zeitschritt 1. Diese beiden Bedingungsbits werden zusammen mit F1 und F2 verwendet, um die SHORT-Maßnahmen zu implementieren.

Die oben beschriebene Pyramide hat eine Basis von MxM und eine Schrumpfung von 2, d.h., daß die Stufe über der Basis M/2 x M/2 PEs hat, und so weiter. Der Steueralgorithmus PYRAMID kann erweitert werden, so daß er eine beliebige Schrumpfung K bearbeiten kann, wobei K eine Potenz von 2 ist, und hmask und vmask durch hmask = hmask & pid0<t> & pid0<t+1> sowie vmask =

vmask & pid1<t> & pid1<t+1> aktualisiert wird. Zusätzlich werden die Maßnahmen der Pyramidenknoten bei dem ungeraden t-Schritt übersprungen.

(P10) Umkehrpyramide

Die Daten fließen von unten nach oben in einer Pyramide, um eine Umwandlung von ikonisch zu symbolisch durchzuführen. Für eine Umwandlung von symbolisch zu ikonisch müssen die Daten in der entgegengesetzten Richtung fließen. Dies wird durch die Umkehrpyramide (R-Pyramide), die aus einem polymorphen Maschennetz gebildet wird, anhand des folgenden Steueralgorithmus erreicht.

```
R-PYRAMID ()

{
int t;/*t ist der Zeitschritt*/
int pid0;/*Reihenposition eines PE*/
int pid1;/*Spaltenposition eines PE*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int mask=M%2;/*Flag zur Bildung der Pyramide*/
int hmask, vmask;

     for(t=0;t<logM;t++){
          hmask = ANDALLBIT (¬mask | pid0);
          vmask = ANDALLBIT (¬mask | pid1);

     /*Zyklus 1*/
          if(¬mask | ¬vmask)
               {SHORT_WE;SHORT_NS;DISABLE;}
          if(hmask && vmask && pid0<logM-t-1> && pid1<logM-t-
1)
               N = MEM2;
          if(hmask && vmask && pid0<logM-t-1> && ¬pid1<logM-t-
1>)
```

```
                MEM2 = S;
            if(hmask && vmask && ¬pid0<logM-t-1> && pid1 <logM-
t-1>)
                NO_ACTION;
            if(hmask && vmask && ¬pid0<logM-t-1> && ¬pid1<logM-
t-1>)
                NO_ACTION;

        /*Zyklus 2*/
            if(¬hmask | ¬vmask)
                {SHORT_WE;SHORT_NS;DISABLE;}
            if(hmask && vmask && pid0<logM-t-1> && pid1<logM-t-
1>)
                {N = MEM1; W = MEM3;}
            if(hmask && vmask && pid0<logM-t-1> && ¬pid1<logM-t-
1>)
                {MEM1 = S; W = MEM2;}
            if(hmask && vmask && ¬pid0<logM-t-1> && pid1<logM-t-
1)
                MEM3 = E;
            if(hmask && vmask && ¬pid0<logM-t-1> && ¬pid1<logM-
t-1>)
                MEM2 = E;

            mask = ASHIFT (mask,1);
        }
}
```

Der Steueralgorithmus für die R_PYRAMID ist ein Umkehrprozeß des Steueralgorithmus PYRAMID und eine Erweiterung von RR_TREE und RC_TREE.

Eine Hälfte der Maschennetzgröße (die "Maske") wird in Register SRM geladen, während pid0 in X und pid1 in Y auf SRX bzw. SRY kopiert werden. Das "UMGEKEHRTE" SRM wird in einer "OR-Operation" mit X verbunden, und dann einer "ANDALLBIT-Operation" unterzogen, um das Flag "hmask" in XANDALL zu

bilden. Auf ähnliche Art und Weise wird "vmask" in YANDALL erzeugt. Zusammen mit diesen beiden Bedingungsbits werden pid0 und pid1 logisch von links nach rechts verschoben, um pid0<logM-t-1> und pid1<logM-t-1> bei Zeitschritt t in BSRX bzw. BSRY zu bilden. In den Figuren 16, 17 und 18 ist die Bildung einer Pyramide in einer 8x8 polymorphen Pyramide dargestellt.

Genau wie bei PYRAMID besteht jeder Schritt des Steueralgorithmus R-PYRAMID aus zwei Zyklen: der erste Zyklus ist eine Zwischenstufe beim Senden von Daten zum NW-Sohn (Daten werden in diesem Zyklus zum NE-Sohn weitergeleitet, während sie im zweiten Zyklus zum NW-Sohn weitergeleitet werden); während im zweiten Zyklus der NE-Sohn Daten zum NW-Sohn weiterleitet, sendet der Vater gleichzeitig Daten zu den NE- und SW-Söhnen.

Die Verbindung für die Nachbarn auf der gleichen Stufe der Pyramide können ähnlich wie bei PYRAMID auch beim Steueralgorithmus R-PYRAMID hergestellt werden. Für den Datenfluß von oben nach unten wird im allgemeinen jedoch keine Kommunikationsverbindung auf der gleichen Stufe hergestellt. Bei Bedarf können ähnliche Maßnahmen wie bei den Zyklen 3 und 4 des Steueralgorithmus PYRAMID verwendet werden.

Die oben beschriebene Pyramide hat eine Basis von MxM und eine Schrumpfung von 2, d.h., daß die Stufe über der Basis M/2 x M/2 PEs hat, und so weiter. Der Steueralgorithmus R-PYRAMID kann erweitert werden, so daß er eine beliebige Schrumpfung K bearbeiten kann, wobei K eine Potenz von 2 ist und SRM durch M/(logK) initialisiert und SRM log K-Bits pro Schritt verschoben wird.

(P11) Würfel

Ein Würfel ist die normale Erweiterung des polymorphen Maschennetzes zu einer 3D-Struktur. Die Anwendung eines Würfels einer 3D-Datenstruktur (z.B. 3D-Bildelement = Volumen

= Voxel) kann analog zur Anwendung des Maschennetzes einer 2D-Datenstruktur (z.B. 2D-Bildelement = Bereich = Pixel) beschrieben werden.

Um einen Würfel im Maschennetz zu bilden, wird die Datenstruktur in der dritten Dimension vertikal aufgeteilt und einem PE zugeordnet, so daß die Kommunikation in der dritten Dimension durch die lokale Speicherkommunikation durchgeführt werden kann, während die Kommunikation im Zusammenhang mit den beiden anderen Dimensionen über das Maschennetz läuft. Die polymorphe Funktion wird bei der Bildung dieser Struktur nicht verwendet, und die Würfelbildung ist daher nicht so neuartig wie die anderen elf Strukturen. Die Würfelstruktur wird jedoch vom polymorphen Maschennetz unterstützt, wobei in der dritten Dimension die Verbindungspins nicht benötigt werden. Da die Einsparung der Pins ins Gewicht fällt (2xMxM Pins insgesamt), ist die Bildung eines Würfels von einem polymorphen Maschennetz wichtig für VLSI-Implementierungen.

Durch das Daten-Slicing kann ein MxMxK-Würfel gebildet werden, wobei K eine ganze Zahl ist, und der Wert von K nur durch den Umfang des lokalen Speichers im PE begrenzt ist.

(P12) Diagonal-Span-Tree (DST)

Ein Diagonal-Span-Tree (DST) ist ein binärer Baum, dessen Blätter die Diagonale des Maschennetzes nur einmal umspannen. Nach dieser Definition hat der DST in einem NxN Maschennetz N Blätter, die jeweils einen diagonalen Knoten besetzen, der als PE (k, N-1-k), k=0 bis N-1 gekennzeichnet ist. In einem Maschennetz sind viele DST möglich. Es wurde ein in Figur 19 gezeigter DST ausgewählt (dargestellt durch einen dreistufigen DST in einem 8x8 Maschennetz), da dieser einfach zu kontrollieren ist.

Aus Figur 19 geht hervor, daß die Wurzel des DST bei PE(0,0) (obere linke Ecke des Maschennetzes) ist. Der linke Sohn der

Wurzel ist vertikal vier Einheiten entfernt (z.B. PE(4,0)), während der rechte Sohn horizontal vier Einheiten entfernt ist (z.B. PE(0,4)).

Die Söhne der zweiten Stufe sind von den entsprechenden Söhnen der ersten Stufe vertikal und horizontal zwei Einheiten entfernt. Daher sind PE(6,0) und PE(2,4) die Söhne von PE(0,4), während PE(4,2) und PE(6,0) die Söhne von PE(4,0) sind.

Bei der Erweiterung auf ähnliche Art und Weise sind alle diagonalen PEs des Maschennetzes die Söhne der dritten Stufe.

Allgemein definiert läßt sich sagen, daß einen UpperLeft DST (oben links DST, ULDST) in einem NxN Maschennetz PE(0,0) als Wurzel und die diagonalen PEs (k, N-1-k), k=0 bis N-1 als Blätter hat. Der linke Sohn von PE(s,t) der i-ten Stufe (i=1 bis logN) ist PE(s+2**(logN - i),t) und der rechte Sohn von PE(s,t) der i-ten Stufe ist PE(s,t+2**(logN-i)).

Der Steueralgorithmus für den ULDST ist nachfolgend aufgeführt.

```
ULDST()

{
int fs=0, fr=0;/*Flag-Send wird zur Bildung des DST
verwendet*/
     /*Flag-Receive ist ein Zwischen-var zur Aktualisierung
von fs*/
int pid0,pid1;
int t;/*t ist der Zeitschritt*/
int M,logM;/*M ist die Seitengröße des Maschennetzes und
logM=log M*/
int treemask=M%2;/*Flag zur Bildung des Baums*/

     if(pid0==0 && pid1==0){
```

```
        fs = 1;fr = 1;}

    for(t=0;t<logM;t++){
        hmask = ANDALLBIT (¬treemask | pid0);
        vmask = ANDALLBIT (¬treemask | pid1);
        if(¬hmask | ¬vmask)
            {SHORT_WE;SHORT_NS;DISABLE;}
        if(hmask | vmask && fr)
            {E = fs; S = fs; fr = 0;}
        if(hmask | vmask && ¬fr)
            {fs = W | N; fr = W | N;}
        treemask = ASHIFT (treemask,1);
    }
}
```

Am Beispiel eines 8x8 polymorphen Maschennetzes kann ein ULDST-Algorithmus wie folgt erklärt werden. Ein Wurzel PE (000,000) wird für den DST ausgewählt, und seine fs und fr werden auf 1 gesetzt. Beim Zeitschritt t=0 sind die Reihen 000 und 100 sowie die Spalten 000 und 100 aktiv, während der Rest deaktiviert ist. Die deaktivierten PEs schließen ihr WE und NS kurz, um einen vorübergehenden Pfad zur Aktualisierung von fs zu errichten. Das aktive PE mit fr=1 sendet seinen fs-Wert zu den E- und S-Nachbarn und setzt fr auf 0, so daß es beim nächsten Zeitschritt kein Sender mehr ist. Die Empfänger (mit fr=0) aktualisieren ihre fs als die Werte von N und W, die einer OR-Operation unterzogen wurden. Die Empfänger aktualisieren ihr fr mit dem Ergebnis, das ebenfalls einer OR-Operation unterzogen wurde. Das PE, das eine 1 von N oder W erhalten hat, wird im nächsten Zeitschritt der Sender sein. Schritt t=0 wählt zwei PEs (PE(000,100) und (100,000)) als Knoten von DST (durch Einstellen ihres fs=1). Darüber hinaus sorgt dieser Schritt dafür, daß sie die neuen Sender sind (durch fr=1), um mehr DST-Knoten im folgenden Schritt zu setzen.

Beim nächsten Schritt erzeugt jeder der beiden Sender zwei

DST-Knoten und zwei Sender auf gleiche Art und Weise. Die neuen Knoten und Sender sind die PEs (000,100), (010,100), (100,010) und (100,000).

Bei t=2 erreicht der Steueralgorithmus die diagonalen PEs; ihr fs wird auf 1 gesetzt, um sie als Teil des DST zu kennzeichnen. Darüber hinaus wird ihr fr als zusätzliche Information auf 1 gesetzt, um sie als diagonale Knoten zu kennzeichnen. Die diagonale Kennzeichnung ist ein Vorteil des DST-Algorithmus. Sie ist für viele Berechnungsarten nützlich, auf die im nachfolgenden Abschnitt näher eingegangen wird.

Um ein DST zu bilden, wird das Flag fs als Bedingung verwendet: PEs mit fs=0 schließen WE- und NS-Pfade kurz, während PEs mit fs=1 MEM zu E und S senden und Daten von W und N empfangen.

Unter Verwendung des Mechanismus im Verbindungssteuerblock wird die "Baummaske" in SRM geladen, fs in F1 und fr in F2. Das UMGEKEHRTE SRM wird mit pid0 in Register X bzw. pid1 in Register Y einer OR-Operation unterzogen; die Ergebnisse aus dieser Operation werden einer "ANDALLBIT-Operation" unterzogen, um "hmask" in XANDALL und "vmask" in YANDALL zu erzeugen. Am Ende jedes Schritts wird SRM arithmetisch um ein Bit nach rechts verschoben. Die SHORT-Maßnahme beruht dann auf F2, XANDALL und YANDALL.

Wenn eine andere Wurzel gewählt wird, jedoch ein ähnlicher Steueralgorithmus verwendet wird, kann ein anderer DST im gleichen polymorphen Maschennetz gebildet werden. In Figur 20 ist ein DST mit einer Wurzel in der rechten unteren Ecke zu sehen. Dieser Baum wird als URDST bezeichnet.

Nachfolgend wird erläutert, wie mit ULDST und LRDST A*X+B*Y+C parallel für jedes Pixel (x,y) in einem Bild berechnet werden kann. Diese Funktion findet in der Computergraphik und Computer Vision breite Anwendung.

ANWENDUNGEN

Neben der bekannten Anwendung eines einfachen Maschennetzes für die Bildverarbeitung sind die nachfolgend beschriebenen Anwendungen des polymorphen Maschennetzes entweder schneller im polymorphen Maschennetz oder im einfachen Maschennetz nicht implementiert. Diese Anwendungen sind in sechs Arten unterteilt.

(1) Teile-und-Herrsche-Berechnung

Hierbei wird ein N Datensatz in zwei Gruppen entsprechend ihrer Eigenschaft unterteilt. Unter Berücksichtigung der gleichen Eigenschaft wird jede Gruppe noch einmal in zwei Untergruppen unterteilt. Dieses Verfahren wird so lange wiederholt, bis jede Gruppe nur noch eine Dateneinheit umfaßt.

Für die Maschennetzverbindung hat diese Art der Berechnung die Komplexität von O(N) oder höher. Bei der Transformation zu Bäumen und Pyramiden ist die Komplexität dieser Berechnungsart O(log N) im polymorphen Maschennetz. Die Beschleunigung für einen Datensatz von N=1024 ist 100:1, d.h. eine Verbesserung von zwei Größenordnungen.

Zu Berechnungen dieser Art gehören Sortieren, Maximum finden, Minimum, k-tes größtes und mittleres. Alle Algorithmen haben die Komplexität O(log N).

(2) Ikonisch-zu-symbolisch Umwandlung

Diese Berechnungsart wird vor allem bei der Computer Vision angewendet und wird auch als Zwischenstufenverarbeitung bezeichnet. Bei einem Bild sind viele Punkte interessant:
(a) wie viele Pixel erfüllen eine bestimmte Eigenschaft;
(b) welche Pixel erfüllen die bestimmte Eigenschaft;
(c) erfüllen EINIGE oder KEINE oder ALLE Pixel die bestimmte

Eigenschaft;

Die oben genannte Eigenschaft kann sein:
(a) gleich mit einem Wert;
(b) größer als ein Wert;
(c) kleiner als ein Wert;
(d) Bedingung arithmetisch und logisch von oben gebildet.

Diese Algorithmen können in O(log N)-Schritten in einem polymorphen Maschennetz durch Baum- und Pyramidenstrukturen berechnet werden. Viel wichtiger und im Gegensatz zu den herkömmlichen festen Strukturen wird nur die Antwort (im Gegensatz zum ganzen Zwischenbild) ausgegeben. Dadurch wird die I/O-Rate erheblich heruntergesetzt; im Extremfall wird nur ein Bit (YES/NO) im Gegensatz zu 1024x1024 Bits (das ganze Bild) ausgegeben.

In bezug auf I/O wurde ein zusätzlicher N-S-Pfad zum Maschennetz hinzugefügt, um gleichzeitig I/O und die Verarbeitung zu unterstützen. Dieser Mechanismus und die damit verbundenen Vorteile finden auch beim polymorphen Maschennetz Anwendung, sind jedoch für diese Erfindung irrelevant.

(3) Statistische Messung

Das polymorphe Maschennetz kann die folgenden Statistiken in O(log N)-Schritten berechnen. Die Statistiken umfassen

(a) mittlere, Varianz-, Standardabweichung;
(b) Bereich, Perimeter, Schwerpunkt;
(c) erstes Moment, zweites Moment und Kreuzmoment;
Punkt (a) bezieht sich auf einen N Datensatz, während Punkt (b) und (c) sich auf ein Bild beziehen.

Die Statistiken bilden die Grundlage für andere Algorithmen. Bei der Computer Vision stellen sie die Basis für Bereichsanalyse und Strukturerkennung dar.

(4) Berechnung von A*x+B*y+C

Um A*x+B*y+C zu berechnen, müssen vier Strukturen vom polymorphen Maschennetz gebildet werden: Oberer-Linker-Diagonal-Span-Tree (ULDST), Unterer-Rechter-DST (LRDST), Reihenbusse und Spaltenbusse. Der ULDST und LRDST müssen zusammen vorhanden sein, um A*X+C und B*Y gleichzeitig zu berechnen, während die Reihen- und Spaltenbusse zusammen vorhanden sind, um die Summe zu berechnen (z.B. A*X+C+B*Y).

Der Algorithmus wird auf bitserielle Art und Weise durchgeführt. Die beiden zusätzlichen Bäume in der Pixelebene können weggelassen werden. Die konstanten Argumente A und B werden vor Beginn der Berechnung an alle PEs weitergeleitet und im Feld A und B gespeichert, wobei die "0" von (logM-1) am Beginn der Felder preemptiert werden. Das Speichern von A und B erfolgt bitumgekehrt, so daß nach dem Zugriff auf die preemptierten "0" auf das niedrigwertigste Bit von A und B zuerst zugegriffen wird. Das konstante Argument C wird zum polymorphen Maschennetz über W der Wurzel des ULDST bei einem Bit pro Zeitschritt geführt, wobei beim niedrigwertigsten Bit begonnen wird. Wenn ULDST als Baum zur Berechnung von A*X+C verwendet wird, hat jedes PE drei Variablen (Summe, Übertrag und Verzögerung). Bei jedem Zeitschritt geht "Summe" in Richtung Osten und "Verzögerung" in Richtung Süden, während jedes PE zwei Operationen durchführt: (a) N bei Feld A hinzufügen, das Übertragbit bei "Übertrag" speichern und (b) N bei "Verzögerung" speichern. Nach logM-Schritten speichern die diagonalen PEs des Maschennetzes (oder die Blätter des ULDST) A*X für die entsprechende Reihe. Auf ähnliche Art und Weise kann die Berechnung von B*Y durch LRDST durchgeführt werden, indem "0" von der Wurzel eingeführt wird, und mit jedem PE "Verzögerung" in Richtung Norden und "Summe" in Richtung Westen geht. Nach logM-Schritten speichern die diagonalen Elemente B*Y für jede entsprechende Spalte.

Nach der Berechnung von A*X+C für Reihen und B*Y für Spalten

geht das polymorphe Maschennetz auf Reihenbusse im WE-Pfad und Spaltenbusse im NS-Pfad über. Jedes PE addiert dann den Wert auf den Reihenbussen zu dem Wert auf den Spaltenbussen, um A*X+B*Y+C auf bitserielle Art und Weise zu erzeugen.

Da ein Ressourcenkonflikt bei der gleichzeitigen Einrichtung der DSTs und Busse vorhanden ist, wird das Ergebnis von A*X+B*Y+C bitseriell bei jedem Zeitschritt gegeben.

(5) Schnelle Zeilendetektion

Wenn A*X+B*Y+C in jedem zweiten Zeitschritt des polymorphen Maschennetzes berechnet werden kann, ist jedes Pixel (X,Y) in einem MxM Bild in der Lage festzustellen, ob es sich auf einer gegebenen Zeile befindet, die von A, B und C festgelegt wird. Wenn alle Zahlen die Länge von K-Bits haben, kann dies in logM+2K Zeitschritten festgestellt werden.

Die schnelle Zeilendetektion ist vor allem in der Computergraphik und Computer Vision sehr nützlich. Bei der Computergraphik dient es vornehmlich zur Anzeige von konvexen Polygonen, zur Erzeugung von Schatten, zum Clipping, zum Zeichnen von Kugeln, zum Berechnen adaptiver Histogrammgleichstellungen, zur Texturabbildung und zum Antialiasing. Bei der Computer Vision eignet es sich vor allem für die Berechnung von Fast Hough Transform zur Erfassung von Zeilen in einem Rauschbild.

(6) Umwandlung symbolischer Daten in ikonische Daten

Bei der in polymorphen Maschennetzen vorhandenen parallelen Hardware ist es von Vorteil, symbolische Daten in ikonische Daten umzuwandeln (dies wird normalerweise nicht in Maschennetzen durchgeführt), so daß die Verarbeitung parallel erfolgen kann. Die oben erwähnte Fast Hough Transform ist ein solches Beispiel. Die nachfolgend beschriebene "Maskenerzeugung" ist eine weitere Klassenanwendung dieser Art.

(6.1) Bandmaskenerzeugung

Eine "Bandmaske" befindet sich innerhalb zweier parlleler Zeilen, von denen eine durch (A, B, C1) und die andere durch (A, B, C2) festgelegt wird. Zur Erzeugung einer "Bandmaske" berechnet jedes PE wie oben beschrieben A*X+B*Y+C1 und A*X+B*Y+C1+(C2-C1). Aus der Berechnung ergibt sich S1 (das Vorzeichen von A*X+B*Y+C1) und S2 (das Vorzeichen von A*X+B*Y+C2), die beide verwendet werden, um festzulegen, ob sich ein Pixel (X,Y) im Band befindet.

Die "Bandmaske" eignet sich insbesondere für Computer Vision, da nur der Bereich von Interesse verarbeitet wird. Die menschliche Sicht benutzt eine andere Strategie zur Erzeugung von Masken. Die "Strategie" ist eine symbolische Information, die jedoch wie beschrieben ikonisch verarbeitet wird.

(6.2) Polygonmaskenerzeugung

Die "Polygonmaske" ist die allgemeine Form einer Bandmaske. Sie besteht aus der Verbindung von P Halbebenen, die jeweils von einer Zeile aus A*X+B*Y+C bestimmt werden. Mit Hilfe der Zeilendetektion können die Vorzeichen S1, S2 bis SP für die entsprechende Zeile berechnet werden. Die Boolesche Verbindung von S1 bis Sp legt fest, daß sich ein Pixel (X,Y) im Polygon befindet.

ZUSAMMENFASSUNG

Im bevorzugten Ausführungsbeispiel der Erfindung werden unter Steuerung des Verbindungssteuermechanismus folgende Transformationen durchgeführt:

Die physische MxM Maschennetzverbindung zu einem linearen Reihen- und Spaltenfeld, das jeweils MxM lang ist.

Die physische MxM Maschennetzverbindung zu M Reihenbäumen, die jeweils M Blätter haben.

Die physische MxM Maschennetzverbindung zu M Spaltenbäumen, die jeweils M Blätter haben.

Die physische MxM Maschennetzverbindung zu einem MxM orthogonalen Baum.

Die physische MxM Maschennetzverbindung zu M Umkehrreihenbäumen, die jeweils M Blätter haben.

Die physische MxM Maschennetzverbindung zu M Umkehrspaltenbäumen, die jeweils M Blätter haben.

Die physische MxM Maschennetzverbindung zu M Reihenbussen, die jeweils M PEs haben.

Die physische MxM Maschennetzverbindung zu M Spaltenbussen, die jeweils M PEs haben.

Die physische MxM Maschennetzverbindung zu einer Pyramide mit einer MxM Basis und einer Schrumpfung K, wobei K eine Potenz von 2 ist.

Die physische MxM Maschennetzverbindung zu einer Umkehrpyramide mit einer MxM Basis und einer Schrumpfung K, wobei K eine Potenz von 2 ist.

Die physische MxM Maschennetzverbindung zu einem MxMxK Würfel, wobei K eine ganze Zahl ist und nur durch den lokalen Speicher des PE begrenzt wird.

Das Verfahren der Erfindung führt unter Programmiersteuerung außerhalb des Verbindungssteuermechanismus folgende Transformationen durch:

Die physische MxM Maschennetzverbindung zu einem DST-Baum, dessen Wurzel sich an einer beliebigen Ecke des Maschennetzes befinden kann. Es können bis zu zwei DST zusammen vorhanden sein, wenn ihre Wurzeln sich an entgegengesetzten Enden einer Diagonale befinden.

Die physische MxM Maschennetzverbindung zu einem MxM orthogonalen Baum in einem O(M**2) Siliziumbereich. Durch das Verfahren der Erfindung kann eine Einsparung in der Größenordnung eines Faktors von O(logM)**2)erzielt werden, wobei M die Seitengröße des orthogonalen Baums ist.

Die Klasse der Teile-und-Herrsche-Algorithmen von linearer O(M) Komplexität zu einer O(log M) Logarithmuskomplexität. Durch das Verfahren der Erfindung kann eine Einsparung von M/logM erzielt werden. Diese Algorithmusklasse wird in der Beschreibung des bevorzugten Ausführungsbeispiels dargestellt.

Die ikonisch-zu-symbolisch Umwandlung (Zwischenstufenverarbeitung) tritt innerhalb des polymorphen Maschennetzes auf, so daß die I/O erheblich reduziert werden kann. Im Extremfall, der bei der Beschreibung des bevorzugten Ausführungsbeispiels der Erfindung näher dargelegt wird, ist eine Reduzierung von sechs Größenordnungen möglich.

Die symbolische Darstellung kann mit Hilfe der oben genannten Strukturen in ikonische Darstellung umgewandelt werden, so daß die Verarbeitung ikonisch mit einer im Maschennetz zur Verfügung stehenden Parallelität durchgeführt werden kann. Diese Funktion erweitert die Fähigkeiten eines Maschennetzes bis in den Bereich der symbolischen Verarbeitung.

Das Maschennetzsystem kann folgendes durchführen:

(a) ikonische Verarbeitung;

(b) Umwandlung von ikonischen Daten in symbolische Daten;

(c) Umwandlung von symbolischen Daten in ikonische Daten und

(d) symbolische Verarbeitung in der entsprechenden ikonischen Äquivalenz.

Das Verfahren der Erfindung ermöglicht die Berechnung von A*x+B*y+C in einem MxM Maschennetz in O(log M) Schritten parallel für jedes Pixel (x,y), wobei A, B und C konstante ganze Zahlen sind.

Mit Hilfe des Verfahrens der Erfindung kann für jedes Pixel (x,y) in einem MxM Bild erfaßt werden, ob sich das Pixel (x,y) (a) auf der Zeile, (b) rechts von der Zeile oder (c) links von der Zeile in einem O(log M) Schritt befindet.

Das Verfahren der Erfindung ermöglicht es, daß bei jedem Pixel (x,y) in einem MxM Bild parallel erfaßt werden kann, ob sich das Pixel innerhalb oder außerhalb eines Bandes befindet, wobei das Band von zwei paralleln Zeilen gebildet wird.

Das Verfahren der Erfindung ermöglicht es, daß bei jedem Pixel (x,y) in einem MxM Bild parallel erfaßt werden kann, ob es sich innerhalb oder außerhalb eines Polygons befindet.

Die Erfindung findet auch auf ein 3D-Maschennetz (physischer Würfel) Anwendung, um eine Dimensionserweiterung der zwölf Strukturen zu erzielen, indem ein Register Z, ein Schieberegister SRZ und ein Flag F3 zum Verbindungssteuermechanismus hinzugefügt wird. (die Dimensionserweiterung eines Würfels ist z.B. ein 4D Hyperwürfel).

Das 3D Bildelement ist ein Voxel. Das Voxel ist das Volumenelement, ähnlich wie das 2D Pixel, das nur über Bereich verfügt. Bei der 3D Erweiterung der Erfindung kann für jedes Voxel (x,y,z) parallel erfaßt werden, ob es sich innerhalb oder außerhalb
(a) eines Bereichs befindet, der von zwei parallelen Ebenen

gebildet wird,
(b) eines Polyhedrons befindet, oder
(c) ob das Voxel in, links oder rechts einer Ebene ist.

Die Erfindung bezieht "polymorph" auf ein physisches Maschennetz über einen "Verbindungssteuermechanismus". Das gleiche Konzept und der gleiche Mechanismus können auch auf andere physisch feste Verbindungen angewendet werden.

Die vom polymorphen Maschennetz gebildeten Strukturen können der Datennatur angepaßt werden, indem die F1 und/oder F2 Register über den Ausgang der ALU beladen werden.

Mit dem polymorphen Maschennetz können auch beliebige Strukturen gebildet werden, indem die F1- und F2-Register durch Befehle oder den Speicher eingestellt werden. Ein Befehl, ein Speicherwert, Zwischenverarbeitungswerte von einem benachbarten Verarbeitungselement und Diagnosedaten innerhalb des Verarbeitungselements sind Beispiele für Systemoperationsparameter, die zur Einstellung der Flag-Register mittels bekannter, hier nicht erläuterter Techniken und Mittel verwendet werden können. Der Verbindungssteuermechanismus umfaßt somit Flag-Registermittel, die als Funktion von Systemoperationsparametern eingestellt werden können, um Steuerdaten bereitzustellen, die für das Einstellen einer neuen Struktur in mindestens einem Strukturregister verwendet werden. Mit Hilfe dieser für den Programmierer zugänglichen Eigenschaft kann dieser Anpassungen an datenbezogene und bedingungsbezogene künftige Ereignismöglichkeiten einstellen. Nach Eintritt eines Ereignisses wird ein Flag-Register eingestellt und eine neue Struktur geholt oder berechnet.

ANSPRÜCHE

1. Ein wiederkonfigurierbares Feldverarbeitungssystem zur Durchführung einer Reihe von Tasks auf Eingangsbilder unter programmierbarer Steuerung, wobei das System verschiedene optimale Konfigurationen bei verschiedenen Zeiten und unter verschiedenen Bedingungen hat, zu denen folgendes gehört:

Systemsteuermittel (3)
ein Feld von Verarbeitungselementen (2), wobei jedes Verarbeitungselement mit einer begrenzten Anzahl von benachbarten Verarbeitungselementen im Feld verbunden ist,

und wobei jedes Verarbeitungselement umfaßt:

einen Speicher (7);

eine ALU (6), die mit dem Speicher verbunden ist; und

ein Verbingungssteuermittel (8), mit einer begrenzten Anzahl von einfachen Verbindungspfaden zur ALU (6) und zu einer begrenzten Anzahl von benachbarten Verarbeitungselementen (2), mit Schaltmitteln (10) zur Bildung selektiver interner und externer Zwischenverbindungen des Verarbeitungselements, um die Kommunikation mit einem nicht benachbarten Zielverarbeitungselement über zahlreich auftretende Verarbeitungselemente entsprechend einer Verbindungssteuerstruktur zu ermöglichen, und Mittel für eine schaltbare Verbindungssteuerstruktur, die Darstellungsmittel (21, 22) umfaßt, die dem Verarbeitungsmittel zahlreiche Strukturbitwerte bereitstellt, die eine Standardmenge von Strukturwerten und eine alternative Menge von Strukturwerten festlegen, sowie ein Strukturwertauswahlmittel (23) zur Auswahl der Standardmenge von Strukturwerten oder zur Auswahl der alternativen Menge von Strukturwerten, wobei die Schaltmittel (10) auf die

ausgewählten Strukturbitwerte reagieren.

2. Ein System nach Anspruch 1, bei dem der Verbindungssteuermechanismus (8) zahlreiche Strukturregister (21, 22), einschließlich einem Standardstrukturregister und einem alternativen Strukturregister umfaßt, sowie einen Crossbar-Schalter (10) mit externen Verbindungen zu benachbarten Verarbeitungselementen und internen Verbindungen zu den zahlreichen Strukturregistern und der ALU (6), und mit Steuerverbindungen zu den zahlreichen Strukturregistern; des weiteren Mittel zur Steuerung des Verbindungssteuermechanismus entsprechend einer Optimierungsstruktur, einschließlich Gate-Schaltmitteln (16-20), die auf die Einstellung im ausgewählten Strukturregister reagieren.

3. Ein System nach Anspruch 2, bei dem der Verbindungssteuermechanismus zwei Strukturregister (21, 22) umfaßt sowie das Strukturauswahlmittel, das ein Strukturregister als binäre Einrichtung (23) auswählt.

4. Ein System nach Anspruch 2, bei dem der Verbindungsstrukturmechanismus Flag-Registermittel (31, 32) umfaßt, und die Flag-Registermittel als Verbindung von Systemoperationsparametern eingestellt werden können, um Steuerdaten bereitzustellen, die für die Einstellung einer neuen Struktur in mindestens einem der Strukturregister benutzt werden können.

FIG. 1

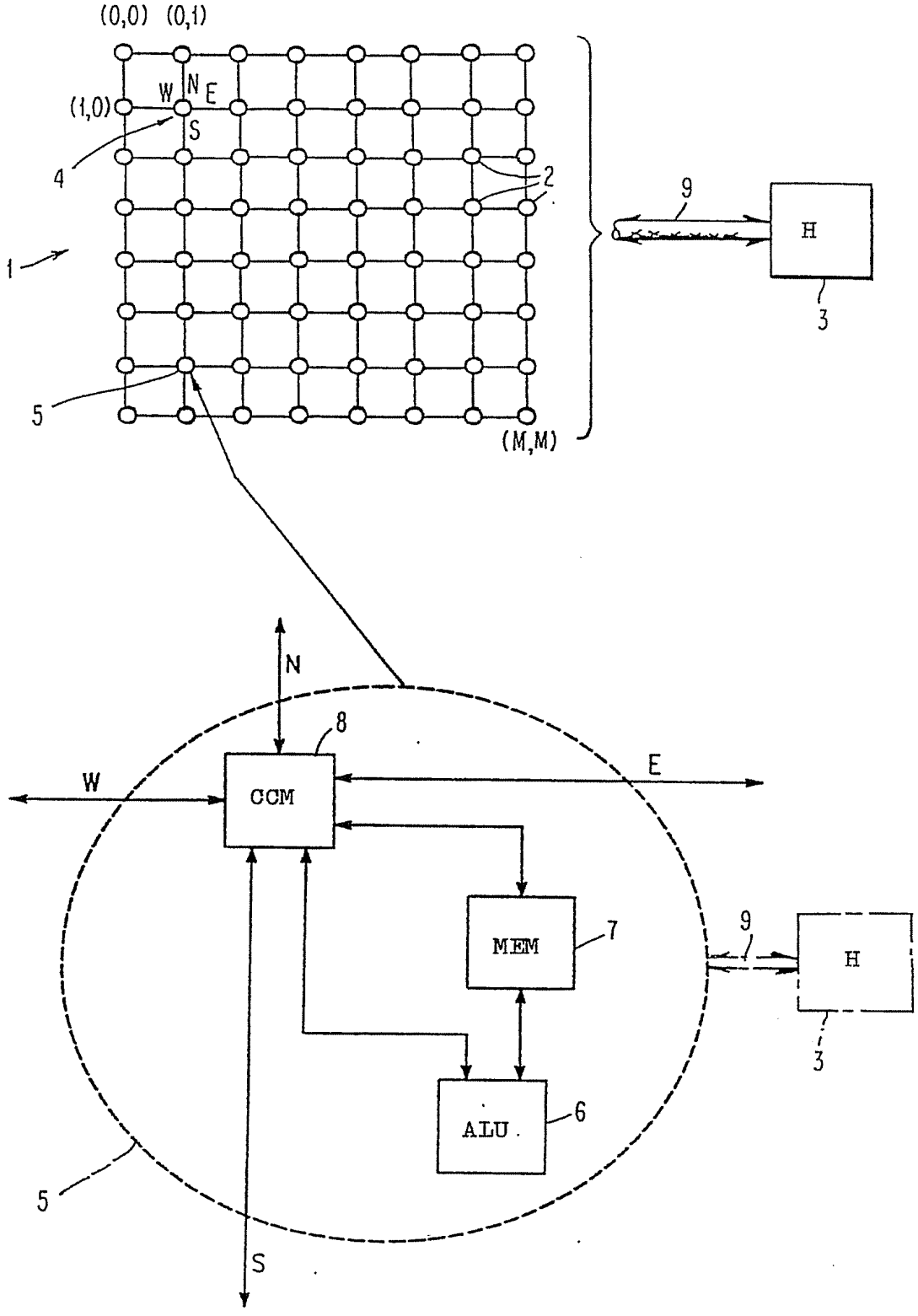


FIG. 2

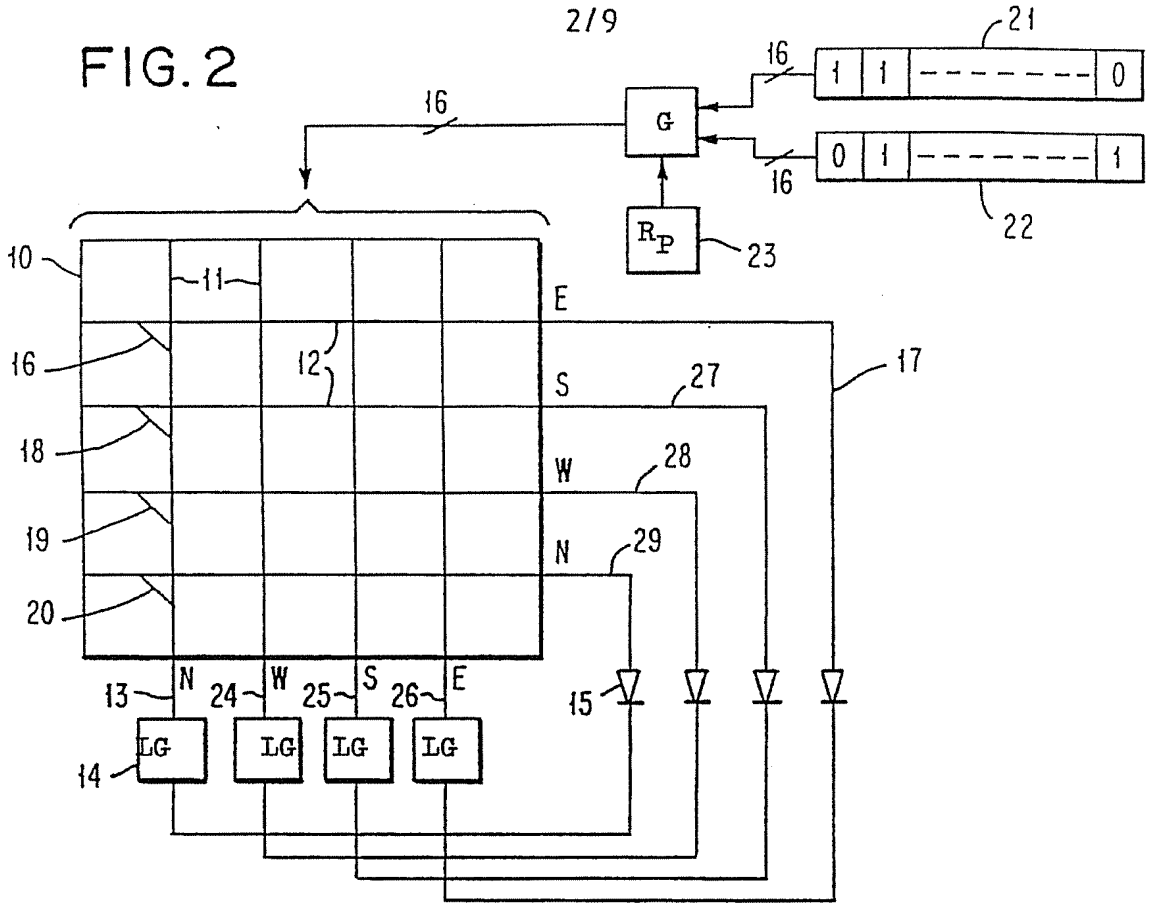

21
16
1
1
0
16
G
0
1
1
16
22
R_P
23
10
11
E
16
12
S
27
17
18
W
28
19
N
29
20
13
N
24
W
25
S
26
E
15
14
LG
LG
LG
LG

FIG. 3

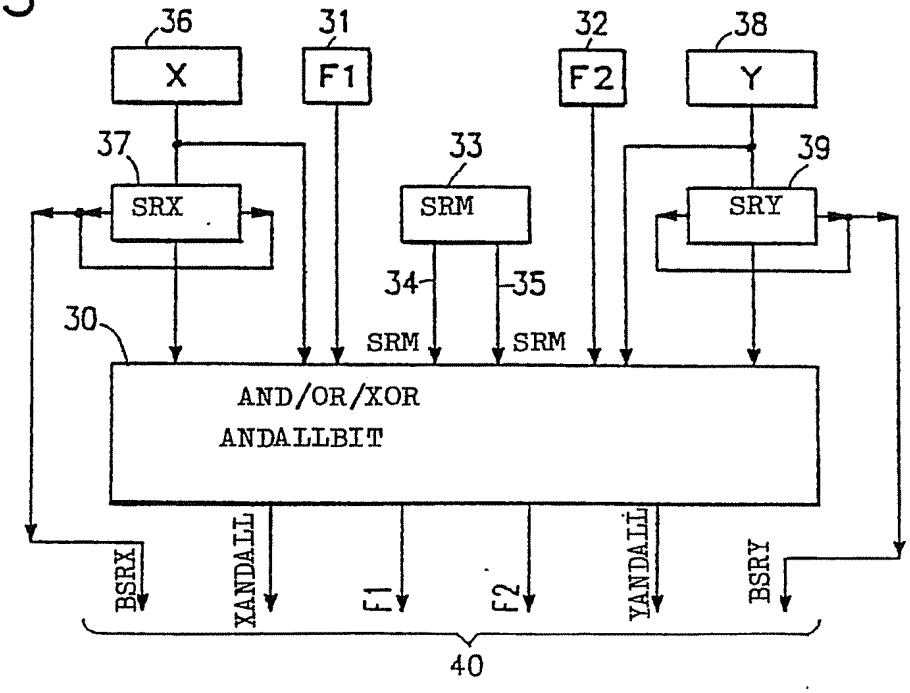

36
X
31
F1
32
F2
38
Y
37
SRX
33
SRM
39
SRY
34
35
30
SRM
SRM
AND/OR/XOR
ANDALLBIT
BSRX
XANDALL
F1
F2
YANDALL
BSRY
40

FIG. 4

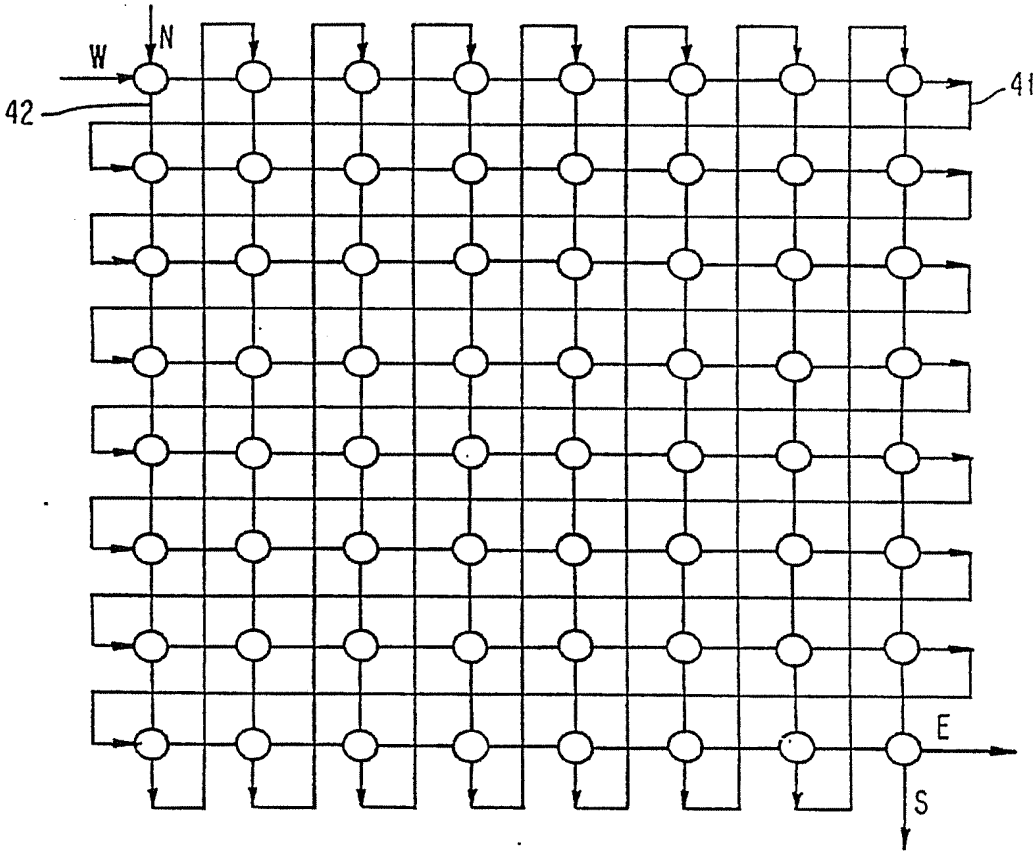


FIG. 5

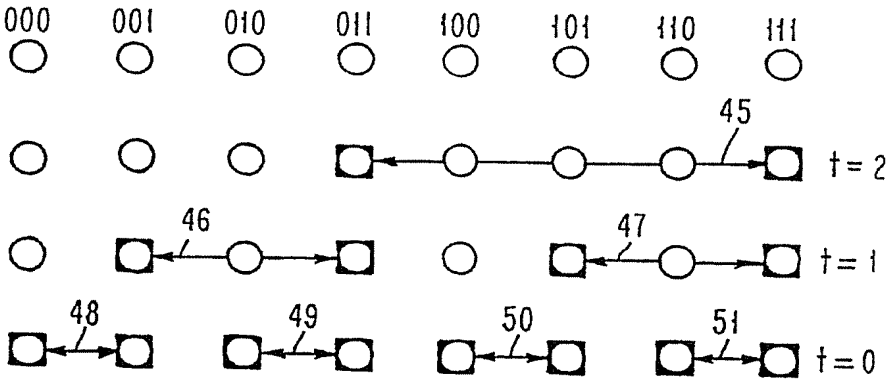

000
001
010
011
100
101
110
111
45
t = 2
46
47
t = 1
48
49
50
51
t = 0

FIG. 6

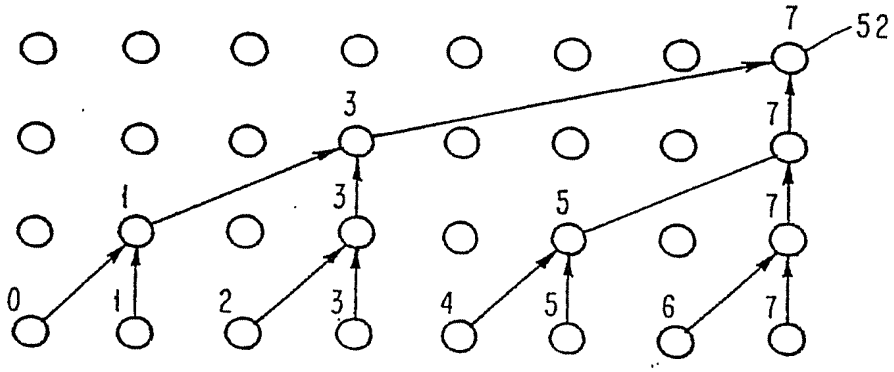

7
52
3
7
1
3
5
7
0
1
2
3
4
5
6
7

FIG. 7

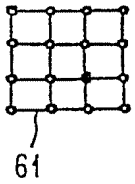


FIG. 9

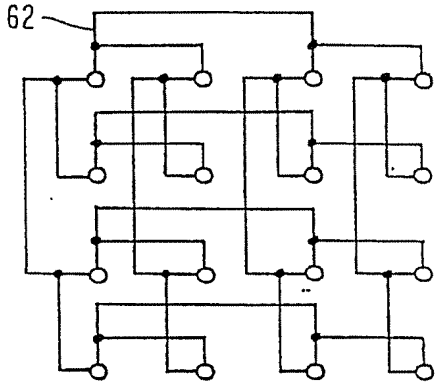


FIG. 8

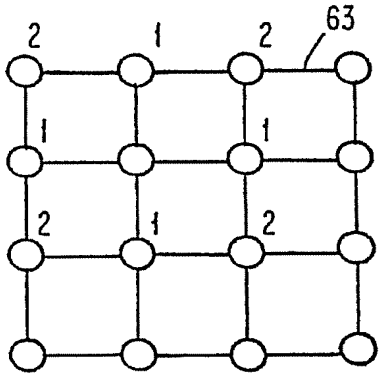


FIG. 10

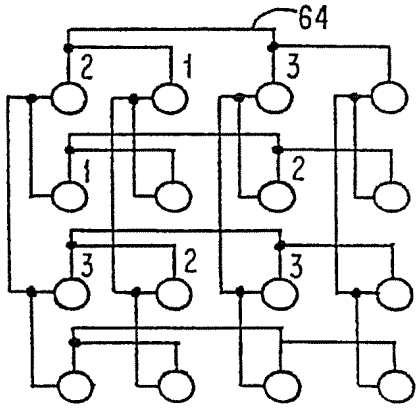


FIG. 11

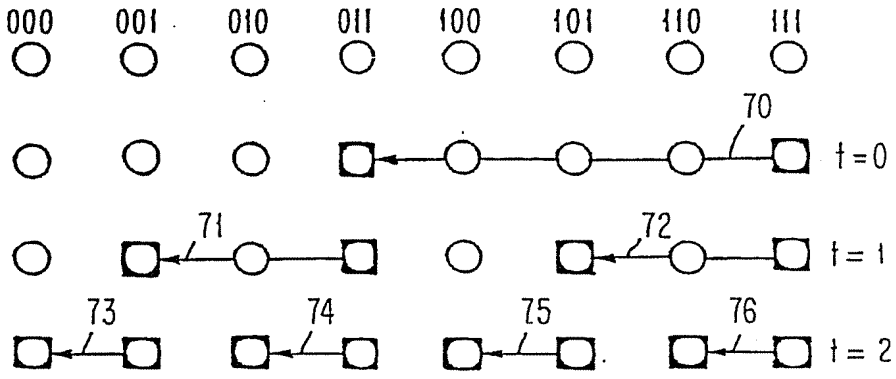


FIG. 12

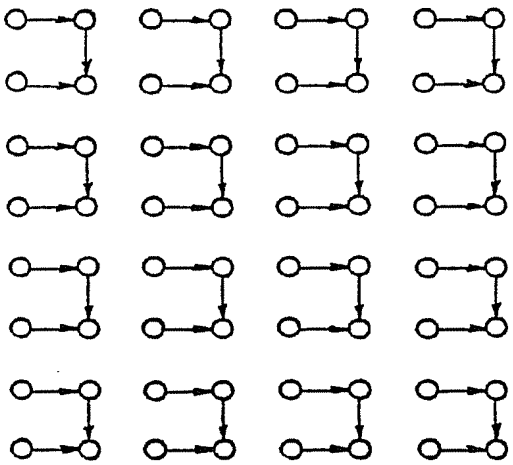


FIG. 14

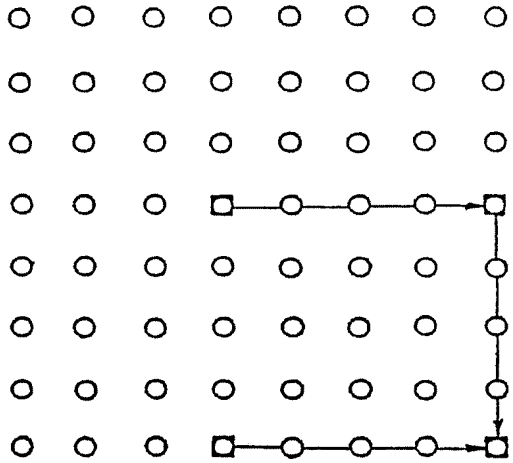


FIG. 13

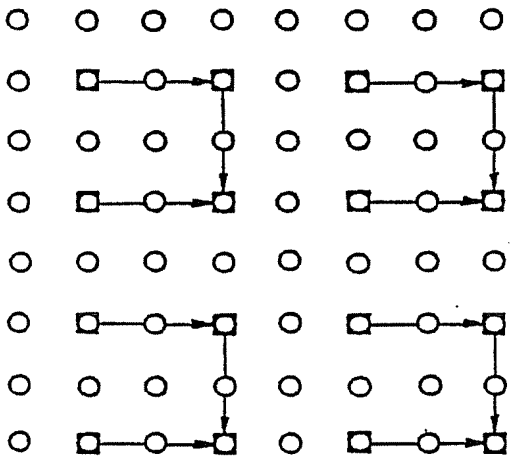


FIG. 15

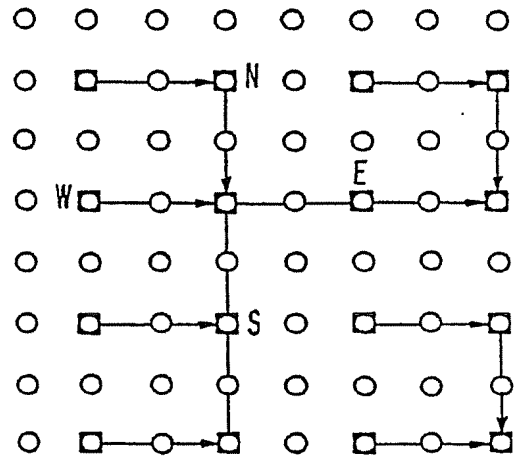


FIG. 16

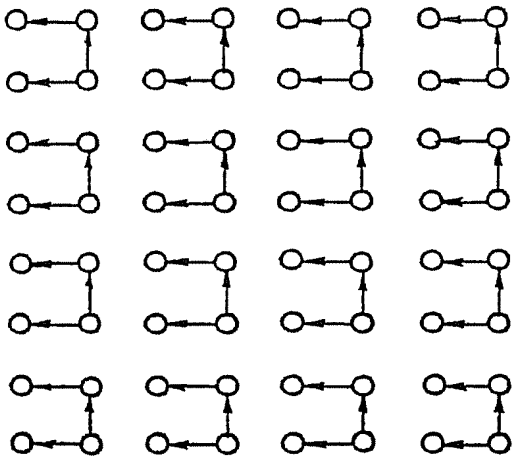


FIG. 17

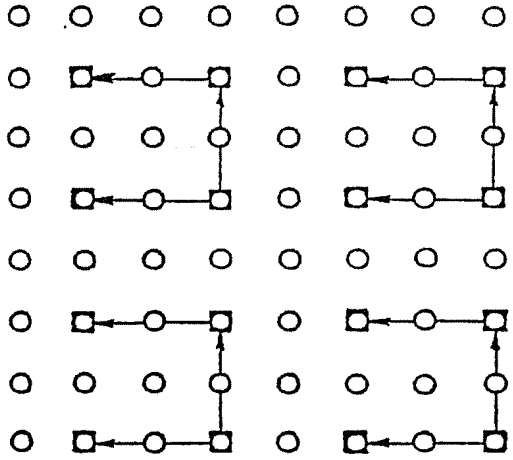


FIG. 18

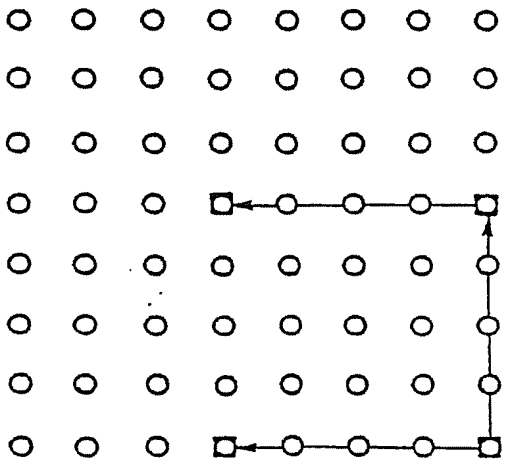


FIG. 19

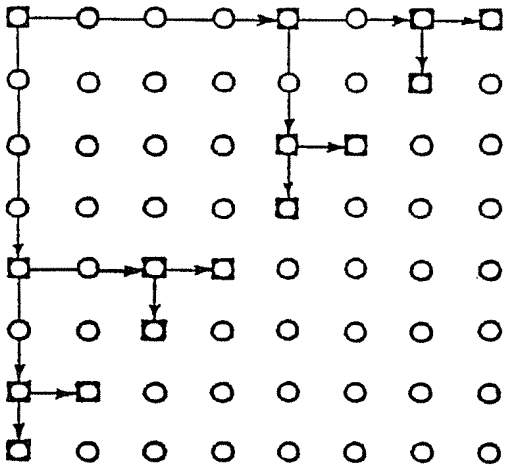


FIG. 20

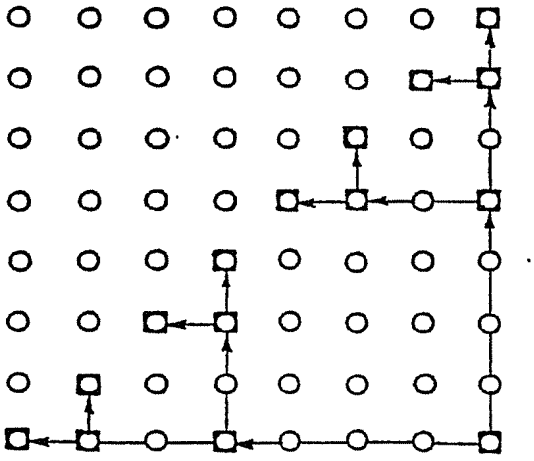
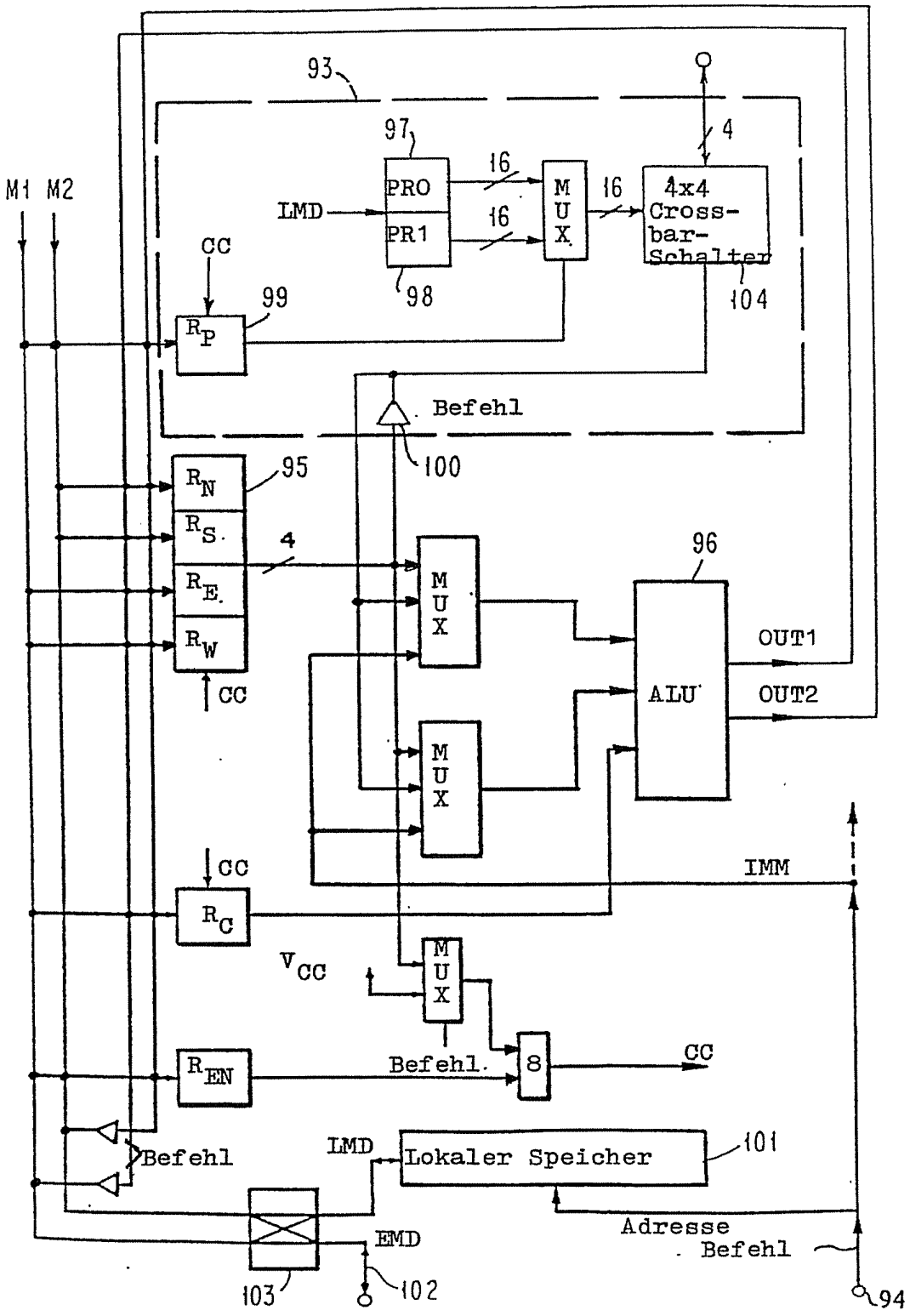


FIG. 21


93
97
4
16
PRO
PR1
16
16
M
U
X
4x4
Cross-
bar-
Schalter
104
98
LMD
CC
99
R_P
M1 M2
Befehl
100
R_N
R_S
R_E
R_W
95
4
CC
96
M
U
X
ALU
OUT1
OUT2
IMM
CC
R_C
V_{CC}
R_{EN}
Befehl
8
CC
Befehl
LMD
Lokaler Speicher
101
EMD
Adresse
Befehl
103
102
94